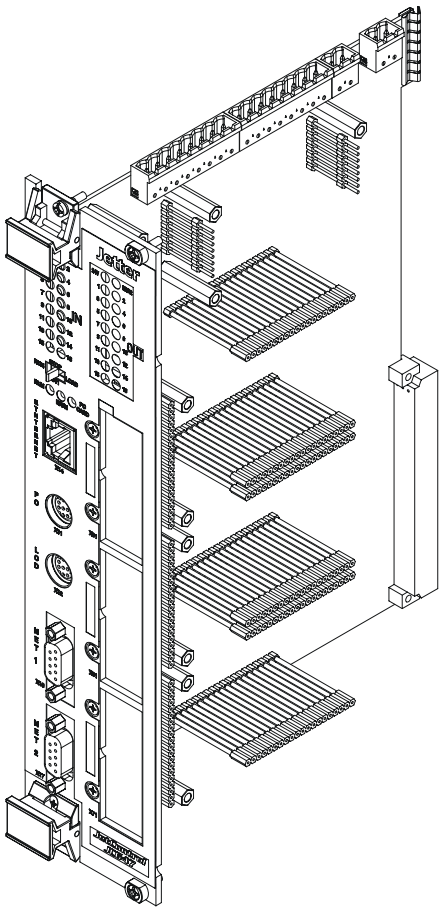
JetControl 647

Controller



JetWeb

Operator's Manual

Jetter
AKTIENGESELLSCHAFT

Article # 60863663 / Revision: 3.50.3 March 2005 / Printed in Germany

Revision 3.50.3

Jetter AG reserves the right to make alterations to its products in the interest of technical progress. These alterations need not be documented in every single case.

This manual and the information contained herein have been compiled with due diligence. However, Jetter AG assumes no liability for printing errors or damages arising from such errors.

The brand names and product names used in this document are trademarks or registered trademarks of the respective title owner.

How to Contact us:

Jetter AG
Gräterstraße 2
D-71642 Ludwigsburg
Germany

Phone - Switchboard:	++49 7141/2550-0
Phone - Sales:	++49 7141/2550-433
Phone - Technical Hotline:	++49 7141 2550 444
Telefax:	++49 7141 2550 425
E-Mail - Sales:	sales@jetter.de
E-Mail - Technical Hotline:	hotline@jetter.de
Internet address:	http://www.jetter.de

This Manual is an Integral Part of the JetWeb Controller JetControl 647:

Model: _____

Serial #: _____

Year of construction: _____

Order #: _____

To be entered by the customer:

Inventory #: _____

Place of operation: _____

© Copyright 2004 by Jetter AG. All rights reserved.

Significance of this Operator's Manual

This manual is an integral part of the JetControl-647 module, and

- must be kept in a way that it is always at hand until the JetControl-647 module will be disposed of.
- If the JetControl-647 module is sold, alienated or loaned, this manual must be handed over.

In any case you encounter difficulties to clearly understand this manual, please contact the manufacturer.
We would appreciate any suggestions and contributions on your part and would ask you to contact us. This will help us to produce manuals that are more user-friendly and to address your wishes and requirements.

This manual contains important information on how to transport, erect, install, operate, maintain and repair the JC-647.
Therefore, the persons carrying out these jobs must carefully read, understand and observe this manual, and especially the safety instructions.

Missing or inadequate knowledge of the manual results in the loss of any claim of liability on part of Jetter AG. Therefore, the operating company is recommended to have the instruction of the persons concerned confirmed in writing.

History

Revision	Comment
2.02	Changes cannot be traced back anymore
2.03	Changes cannot be traced back anymore
2.04	Changes cannot be traced back anymore
3.50.1	For changes refer to Revision 3.501, Appendix B
3.50.2	See “Recent Revisions” on page 363.

Description of Symbols

Danger

This sign is to indicate a possible impending danger of serious physical damage or death.

Caution

This sign is to indicate a possible impending danger of light physical damage.
This sign is also to warn you of material damage.

Important

This sign is to indicate a possible impending situation which might bring damage to the product or to its surroundings.
It also identifies requirements necessary to ensure faultless operation.

Note

You will be informed of various possible applications and will receive further useful suggestions.
It also gives you words of advice on how to efficiently use hardware and software in order to avoid unnecessary efforts.

· / -

Enumerations are marked by full stops, strokes or scores.

Operating instructions are marked by this arrow.

Automatically running processes or results to be achieved are marked by this arrow.

PC and HMI keys.

This symbol informs you of additional references (data sheets, literature, etc.) associated with the given subject, product, etc. It also helps you to find your way around this manual.

Contents

1 Safety Instructions 15

1.1 Ensure Your Own Safety 17

1.2 Instructions on EMI 19

2 Overview of Functions 21

2.1 Operating System Release 21

2.2 Functions 21

2.3 Register Areas 21

2.4 Program Memory and Data Formats 22

2.5 Functional Diagram 23

3 Design of the JC-647 25

4 Operating Conditions 27

5 JX6 Rack 33

5.1 Physical Dimensions 33

5.1.1 JX6-Basis-1 with One Slot 33
5.1.2 JX6-Basis-2 with Two Slots 34
5.1.3 JX6-Basis-4 with Four Slots 35
5.1.4 JX6-Basis-8 with Eight Slots 36

5.2 Numbering of Slots 37

6 Installation Guide 39

6.1 Deinstalling the JC-647 Module 39

6.2 Installing the JC-647 Module 41

6.3 Using the keying pins 42

7 Submodules for Sockets # 1 through 3 45

7.1 Submodule types 45

7.2 Submodule Type Registers 47

7.3 Register Areas of Submodules 49

7.3.1 Socket 1 49
7.3.2 Socket 2 49
7.3.3 Socket 3 49

7.4 JX6-AD8 Submodule 50

7.4.1 Description of Connections 51
7.4.2 Overview of Registers 52

7.5 Submodule JX6-DA4 54

7.5.1 Description of Connections 55
7.5.2 Overview of Registers 56
7.6 Submodule JX6-SV 57
7.6.1 Description of Connections 57
7.6.2 Overview of Registers 58
8 PC-Card 59
8.1 Hardware 59
8.2 Software 59
9 RUN - STOP - LOAD Switch 61
10 Power Supply 63
10.1 Requirements 63
10.2 Description of Connections 64
11 Digital Inputs 65
11.1 Technical Data 65
11.2 Meaning of LEDs 66
11.3 Description of Connections 67
11.4 Numbering System of Digital Inputs 68
12 Submodule JX6-I16 69
12.1 Technical Data 69
12.2 Meaning of LEDs 71
12.3 Description of Connections 72
12.4 Numbering System of Digital Inputs 73
13 Submodule JX6-O16 75
13.1 Technical Data 75
13.2 Meaning of LEDs 77
13.3 Description of Connections 78
13.4 Numbering system of digital outputs 79
14 Counters 81
14.1 Overview of Functions 81
14.2 Meaning of Registers 82
15 Serial Interfaces 85
15.1 Programming Interface PC 86
15.1.1 Description of Connections 86
15.2 JetSym Settings 88

15.3 HMI LCD Interface 89
15.3.1 HMI Cable JN-DK-Xm 89
15.3.2 HMI Cable KAY_0386-xxxx 90
15.3.3 HMI Cable KAY_0533-0025 92
15.4 Programming Interface via JETWay-H 94
15.4.1 Selecting RS-232 or JETWay-H 94
15.4.2 Description of Connections 95
15.4.3 JETWay-H PC Card 96
15.5 Network interface (JETWay-R) 98
15.6 User-Programmable Interface 99
16 Ethernet Connection 101
16.1 Connecting cables to the RJ45 Jack 101
16.2 Connection between JetControl 647 and PC 102
16.3 Connection between JetControl, Switch and PC 102
16.4 Connection via Several Switches 103
17 Software Programming 105
17.1 Instructions 105
17.2 Access to Inputs or Outputs 106
17.3 Access to Flags 107
17.3.1 User Flags 107
17.3.2 Programming by Means of Flags 107
17.4 Access to Registers 109
17.4.1 Programming by Means of Registers 109
17.4.2 Calculations by Means of Registers 113
17.4.3 Accessing Digital Inputs Using Registers 124
17.4.4 Accessing Digital Outputs Using Registers 130
17.5 Positioning Instructions 136
18 HMIs, Operator Guidance 137
18.1 Technical Data 137
18.2 Description of Connections 138
18.3 Multi-Display Mode 139
18.4 Programming the HMIs 141
18.4.1 Displaying Texts 141
18.4.2 Displaying Registers 144
18.4.3 USER_INPUT 145
18.4.4 USER_INPUT: Default Value 146

19 Submodules JX6-SB, and JX6-SB-I 147

19.1 Technical Data 147

19.1.1 Operating Mode "Master-Slave JX6-SB" 148
19.1.2 Operating Mode "Master-Master" 152

19.2 Operating Modes 154

19.3 System Bus - Baud Rate 155

19.4 Diagnostic Indicators (LEDs) 155

19.5 Description of Connections 157

19.5.1 JETTER System Bus Cable Specification 158

19.6 Firmware - Master-Slave JX6-SB(-I) 160

19.6.1 Overview of Registers 160
19.6.2 IO Range 169

20 Network Operation JetWay 171

20.1 JETWay-H: JETTER Data Highway 171

20.2 JETWay-R: Process Level 172

20.3 N_SEND_REGISTER and N_GET_REGISTER 174

20.3.1 N_SEND_REGISTER and N_GET_REGISTER 174
20.3.2 N_GET_REGISTER 175

20.4 Network Operation with 50000-Numbers 175

21 Network Operation via Ethernet 177

21.1 Setting the IP-Address 177

21.2 N_COPY_TO, and N_COPY_FROM 179

21.2.1 N_COPY_TO 179
21.2.2 N_COPY_FROM 182

22 Introduction of JetControl Web Functions 185

22.1 Initialization of Web Functions 186

22.2 The FTP Server 186

22.2.1 Log-In 186
22.2.2 Key Instructions 187

22.3 HTTP Server 187

22.3.1 Supported File Types 187
22.3.2 Serverside Includes 187

22.4 The E-Mail Client 188

23 File System 189

23.1 Properties 189
23.2 User Administration 189
23.2.1 Flash Disk Lock File 190
23.2.2 Key Names File 191
23.2.3 User File 192
23.3 System directories / files 193
23.3.1 The files “/System/flashdisklock.ini”, “/System/keys.ini”, and “/System/users.ini” 193
23.3.2 The file "/System/flashdiskinfo.txt" 194
23.4 Formatting the Flash Disk 195
23.4.1 Formatting by means of Registers 195
24 FTP Server 197
24.1 Log-In 197
24.2 Supported Commands 198
24.3 Example: Windows FTP Client 199
25 E-Mail 201
25.1 The E-Mail INI file 201
25.2 Structure of the E-Mail Template File 202
25.3 Name of the E-Mail Template File 203
25.4 Sending E-Mails 204
25.5 The E-Mail Default Template File 205
25.6 Transmission of Controller Values 206
25.6.1 Format Definition 206
25.6.2 Register - Text Variable 207
25.6.3 Register - Bit Variable 208
25.6.4 Flags 209
25.6.5 Inputs 210
25.6.6 Outputs 211
25.6.7 Indirect Access 212
25.7 Peak Values 213
25.8 E-Mail Tracking Register 213
26 HTTP Server 215
26.1 Supported File Types 215
26.2 Serverside Includes 215
26.2.1 Name Space Tag 216
26.2.2 Data Tag 216
26.2.3 Example Page 223

26.3 Visualization via Internet browser 224
27 RemoteScan 225
27.1 Special/System Functions 225
27.1.1 Configuring RemoteScan 225
27.1.2 Starting RemoteScan 228
27.1.3 Stopping RemoteScan 228
27.2 Registers 228
28 Modbus/TCP 229
28.1 Modbus/TCP Server 229
28.1.1 Supported Commands - Class 0 229
28.1.2 Supported Commands - Class 1 229
28.1.3 Supported Commands - Class 2 231
28.2 Modbus/TCP Client 232
28.2.1 RemoteScan 232
28.2.2 Special/System Functions 233
29 Data Files 237
29.1 Special/System Functions 237
29.1.1 Implementation 237
29.1.2 File Names 237
29.1.3 Saving values – Creating a file 238
29.1.4 Saving values – Appending to a file 239
29.1.5 Reading values from a file 240
29.1.6 Deleting a file 241
29.2 Registers 241
29.3 File Format 242
30 Special Flags 243
31 Overview of Special Registers 251
32 Special Registers 271
32.1 Operating System (Error) Messages 271
32.2 Computing Speed 275
32.3 Task Control 276
32.4 Controlling HMIs (LCDs) 278
32.4.1 Entering Fixed-Point Numbers 278
32.4.2 Displaying a Suggested Value 281
32.4.3 Entering Fixed-Point Numbers 282
32.4.4 Indication 285
32.4.5 Restrictions for User Input 296
32.4.6 Mapping of LEDs on HMIs 297
32.4.7 Baud rate of HMI (LCD) 300

32.5 Ethernet Control 301
32.5.1 Addressing 301
32.5.2 Times 301
32.5.3 Error Messages 302
32.6 Web-Function Initialization 303
32.7 E-mail Tunction 304
32.8 RemoteScan Function 305
32.9 Data Files 306
32.10 Instruction Register 307
32.11 Controlling Network # 1 308
32.11.1 Address, baud rate 308
32.11.2 Times 310
32.11.3 Error Messages 311
32.12 Controlling Network # 2 312
32.12.1 Configuration of network # 2 312
32.12.2 JETWay 313
32.12.3 User-Programmable Interface 315
32.13 Control of the Programming Interface (PC) 318
32.14 Interface Activity Monitoring 319
32.15 Status of the Application Program 324
32.16 Timer Registers 326
32.17 General Registers 327
32.18 Specifying the floating point register array 329
33 Special Functions 331
33.1 General Information 331
33.2 Function Types (Overview) 332
33.3 Format Conversion 333
33.4 Arithmetic Functions 334
33.5 Sorting data 337
33.6 Modbus RTU CRC-Checksum 338
33.7 Modbus/TCP 340
33.8 RemoteScan 341
33.9 Data Files 342
33.10 E-Mail 343
34 Sorting data 345
34.1 Special/System Function 50 345

34.2 JetSym Sample Program 347

35 Oscilloscope of JetSym 349

36 User-programmable Interface 351

36.1 Description of Connections 351

36.2 Register Description 351

36.3 Program example 351

36.3.1 Program Listing 351
36.3.2 Symbol Listing 352

37 Real-Time Clock 353

38 Programming Language JetSym ST 357

List of Appendices

Appendix A: Downloading the Operating System 361
Appendix B: Recent Revisions 363
Appendix C: Glossary 368
Appendix D: List of Abbreviations 372
Appendix E: Register Overview by Numeric Order 375
Appendix F: List of Illustrations 385
Appendix G: Index 387

1 Safety Instructions

The JetControl-647 module complies with the safety regulations and standards in effect. Special emphasis was given to the safety of the users.

Of course, the user should adhere to the following regulations:

- relevant accident prevention regulations;
- accepted safety rules;
- EC guidelines and other country-specific regulations.

Usage as Agreed Upon

Usage as agreed upon includes operation in accordance with the operating instructions.
The JetControl-647 module is used to control machinery, such as conveyors, production machines, and handling machines.
The supply voltage of the JetC-647 module is DC 24 V. This operating voltage is classified as SELV (Safety Extra Low Voltage). The JetControl-647 module is therefore not subject to the EU Low Voltage Directive.

The JC-647 module is operated in slot 1 of the JX6 rack.

The JC-647 module may only be operated within the limits of the stated data.

Usage Other Than Agreed Upon

The JC-647 module must not be used in technical systems which to a high degree have to be fail-safe, e.g. ropeways and aeroplanes.l

If the JC-647 module is to be run under surrounding conditions, which differ from the conditions mentioned in chapter 4: "Operating Conditions", page 27, the manufacturer is to be contacted beforehand.

Who is Permitted to Operate the JC-647 Module?

Only instructed, trained and authorised persons are permitted to operate the JC-647 module.

Transport:	Only by personnel with knowledge in handling electrostatically sensitive components.
Installation:	Only by specialists with training in electrical engineering.
Commissioning:	Only by specialists with extensive knowledge of and experience with electrical engineering / drive technology.

Modifications and Alterations to the Module

Due to safety reasons, no modifications and alterations to the JC-647 module and its functions are allowed. Any modifications to the JC-647 module not expressly authorized by the manufacturer will result in a loss of any liability claims to Jetter AG.

The original parts are specifically designed for the JC-647 module. Parts and equipment of other manufacturers are not tested on our part, and are, therefore, not released by us. The installation of such parts may impair the safety and the proper functioning of the JC-647 module.

Any liability on the part of Jetter AG for any damages resulting from the use of non original parts and equipment is excluded.

Maintaining the JC-647 Module

The JC-647 module is not maintenance-free. It is equipped with a battery which has to be replaced at regular intervals. The designed service life of the original battery is > 10 years. The battery condition can be checked in the following way:

Special Flag 2146 = 1:	Battery for real-time clock is ok	
Special Flag 2147 = 1:	Battery for register-RAM is almost flat	
or		
Special register 61478	Bit 2 = 1:	Battery for real-time clock is ok
	Bit 3 = 1:	Battery for register RAM is almost flat

If the battery needs to be replaced, please send the JC-647 controller to Jetter AG.

Note!

The real-time clock and the RAM for application registers are backed up by the same battery.

Repairing the JC-647 Module

The JC-647 module must not be repaired by the operator. The JC-647 module does not contain any servicable parts that could be repaired by the operator.
The JC-647 module must be sent to Jetter AG for repair.

Decommissioning and Disposing of the JC-647 Module

Decommissioning and disposal of the JC-647 module are subject to the environmental legislation of the respective country in effect for the operator's premises.

1.1 Ensure Your Own Safety

- Disconnect the JC-647 module from the mains to carry out maintenance work. By doing so, you will prevent accidents resulting from electric voltage and moving parts.
- Safety and protective devices, e.g. the barrier and cover of the terminal box must never be shunted or by-passed.
- Dismantled protective equipment must be reattached prior to commissioning and checked for proper functioning.

Malfunctions

- Malfunctions or other damages are to be reported to an authorised person at once.
- Safeguard the JC-647 module against misuse or accidental use.

Information Signs and Labels

- Writings, information signs, and labels always have to be observed and kept readable.
- Damaged or unreadable information signs and labels are to be exchanged.

Earthing procedure

- Screw down the JX6 housing to a highly conductive and earthed mounting plate.

- An earth-terminal screw with a M4 thread is located on the top face of the JX6 housing.
  This earthing screw must electrically be connected to a PE terminal in the electric cabinet by means of a PE conductor (conductor area: 1,5 mm^2, color: green-yellow) (refer to Fig. 1).

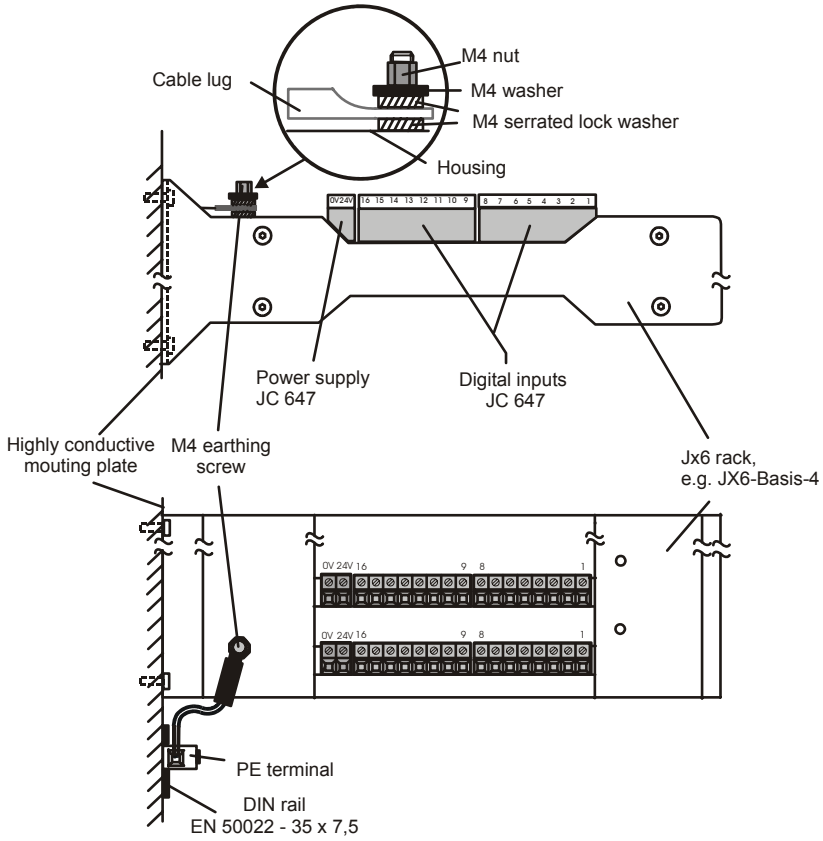


Fig. 1: Earthing, JX6 rack

1.2 Instructions on EMI

In terms of emitted interference the JC-647 module is a product belonging to EMC class A. This module may cause radio interferences in residential areas. This module is operated at the operator's own risk.

The noise immunity of a system corresponds to the weakest component of the system. For this reason, correct wiring and shielding of cables is of paramount importantance.

Important!

Measures for increasing immunity to interference:

- Attach the JC-647 module firmly to the JX6 housing (see Fig. 10).
- Follow the instructions given in Application Note 016 "EMC-Compatible Installation of the Electric Cabinet" published by Jetter AG.

The following instructions are excerpts from Application Note 016:

- It is of great importance that the JX6 rack is screwed down to a highly conductive mounting plate.
- On principle, physical separation should be maintained between signal and voltage lines. We recommend spacings greater than 20 cm. Cables and lines should cross each other at an angle of 90°.
- Shield cables on both ends.
- Unshielded wire ends of shielded cables should be as short as possible.
- The entire shield must be drawn behind the isolation, and then be clamped under an earthed strain relief with the greatest possible surface area.

When male connectors are used:

- Only use metallized connectors, e.g. SUB-D with metallized housing. Make sure that the strain relief is directly connected with the housing here as well (see Fig. 2).

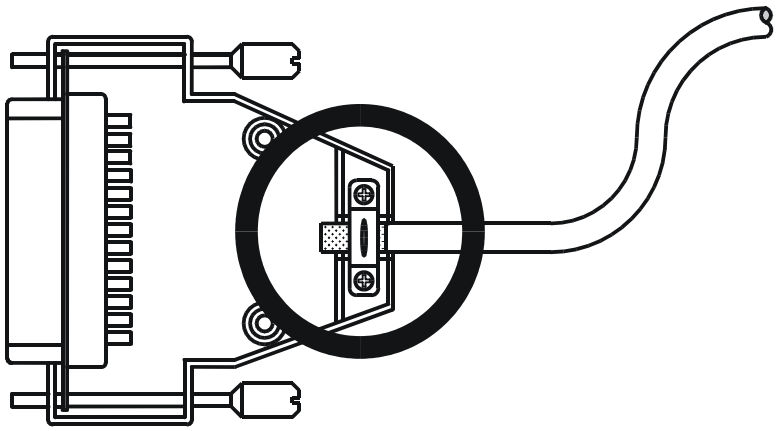


Fig. 2: Shielding of SUB-D connectors in conformity with EMC standards.

Important!

To avoid malfunctions the following must be ensured:

- Adhere to the measures for increasing immunity to interference. Compliance with these measures will also prevent malfunctions.

2 Overview of Functions

2.1 Operating System Release

Note!

This manual gives a description of functions provided by release 3.50 of the JetControl 647 operating system and release 2.3.1 of the programming software JetSym.

2.2 Functions

- CPU, 256 kByte program memory, 20480 + 196608 registers
- 1 serial programming interface PC
- 1 LCD interface
- 1 network interface JetWay 1 (JetWay-H, -R)
- 1 JetWay 2 / user-programmable interface (JetWay-H, -R)
- 3 submodule sockets (for JX6-AD8, JX6-DA4, JX6-SV, JX6-PRN, JX6-INT, JX6-SB, JX6-SB-I, JX6-ASI-INT1, JX6-PROFI, JX6-VCS)
- 16 digital inputs
- 16 digital inputs or 16 digital outputs (optional)
- 4 counters
- Real-time clock
- Operating system download via serial PC-interface or via PC-card can be realized by means of an update.

2.3 Register Areas

Application registers (SRAM1)	0 .. 20479
New application registers (SRAM2)	196608 .. 393215
PC-Card	393216 .. 2490367
System and special registers	61440 .. 64999
Floating Point Registers	62208 .. 62463
Combined inputs	62464 .. 62655
Combined outputs	62720 .. 62911
Real-time clock	62912 .. 62928
Area for non-intelligent submodules Socket # 1 Socket # 2 Socket # 3	 63300 .. 63399 63400 .. 63499 63500 .. 63599

Slave registers for Socket # 2 Socket # 3 Socket # 4 Socket # 5 Socket # 6 Socket # 7 Socket # 8	 121000 .. 124999 131000 .. 134999 141000 .. 144999 151000 .. 154999 161000 .. 164999 171000 .. 174999 181000 .. 184999
Area for intelligent submodules	1xyp00 .. 1xyp63 Module slot x: 1 .. 8 Submodule socket y: 1 .. 3 Port p: 1 .. 4
Combined Inputs / Outputs RemoteScan JX6-INT, JX6-SB(-I), JX6-PROFI	111300 .. 183463
Non-intelligent JX2-IO modules, Lumberg modules und Smart I/O modules connected to JC-647	3102000 .. 3307999
JX2-Slave module and JetMove connected to JC-647	3112100 .. 3319999

2.4 Program Memory and Data Formats

Program memory	256 KByte Flash-EPROM
User register 32 bits	197,607 battery buffered RAM
32-bit registers, either application or floating point registers	19,480 battery-backed RAM
Floating point register 32 bits	255 battery-backed RAM
Data format	**32-bit integer:** -2.147.483.648 .. +2.147.483.647 **32-bit floating point:** +/- ($8{,}43 \cdot 10^{-37}$ through $3{,}38 \cdot 10^{38}$)
Internal intermediate results	32 bits

2.5 Functional Diagram

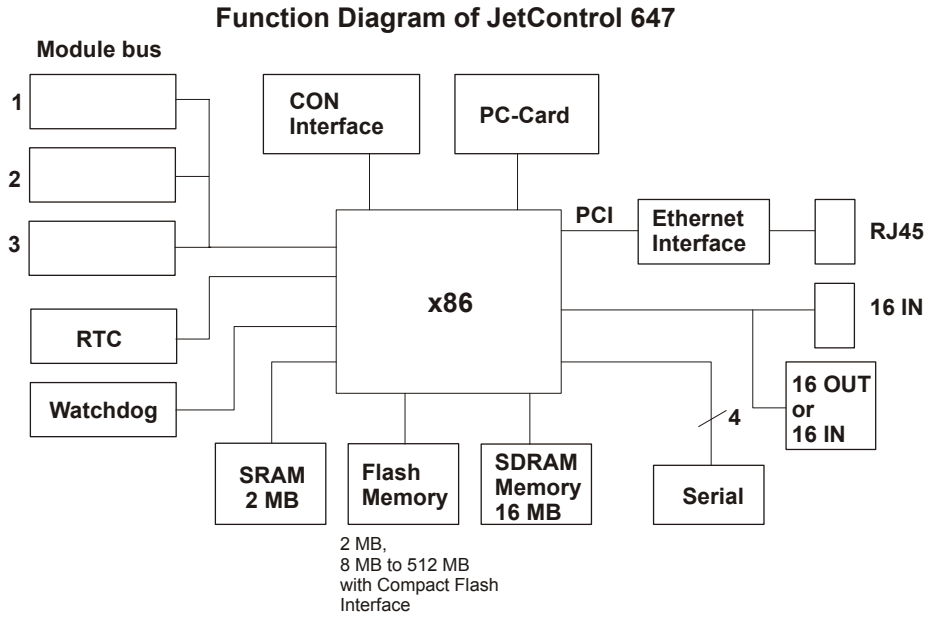
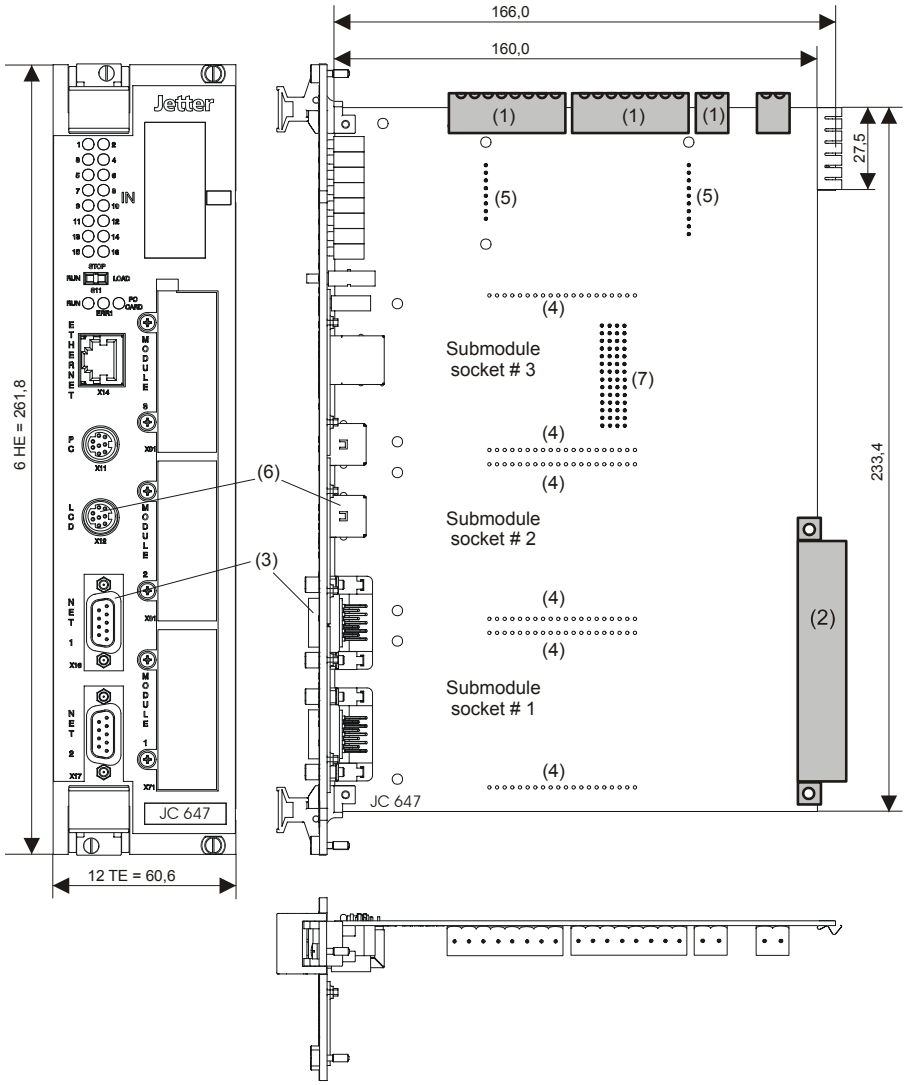


Fig. 3: Functional Diagram

3 Design of the JC-647



(1): Connector in the COMBICON basic housing RM 5.08
(2): Plug connector type C, 96-pin
(3): Sub-D connector (female)
(4): Socket connector for submodule sockets 1 to 3
(5): Plug connector for submodule JX6-I16 or JX6-O16
(6): Mini DIN jack/connector
(7): Plug connector für PC-Card

Fig. 4: Front, Side and Top Views of the JC-647

Type	
Dimensions (H x W x D in mm)	261,8 x 60,6 x 185
JC-647 with 16 digital inputs	Can be inserted into module slot # 1 of the JX6-rack

4 Operating Conditions

Operating Parameters of JC-647 mounted in JX6-Basis-x rack

Power Rating

Features	Value(s) (depending on the product)	Preferable reference standard(s) (depending on the product)
Power Supply Unit Requirements	DC 24 V (20 .. 30 V) Residual ripple < 5 % filtered	
Power consumption without LCD, options and digital outputs	approx. 13 W	
Voltage dips	Duration of voltage dips <= 10 ms Severity level PS2	DIN EN 61131-2

Environment

Features	Value(s) (depending on the product)	Preferable reference standard(s) (depending on the product)
Operating Temperature Range	0 °C - +50 °C	-
Storage Temperature Range	-25 °C - +70 °C	DIN EN 61131-2 DIN EN 60068-2-1 DIN EN 60068-2-2
Air humidity	5 % - 95 %, Non condensing	DIN EN 61131-2
Pollution Degree	2	DIN EN 61131-2
Corrosion immunity/ chemical resistance	No special protection against corrosion. Ambient air must be free from higher concentrations of acids, alcaline solutions, corrosive agents, salts, metal vapours, or other corrosive or electrocon-ductive contaminants	General specification
Atmospheric pressure	2000 m	DIN EN 61131-2

Mechanical Parameters

Features	Value(s) (depending on the product)	Preferable reference standard(s) (depending on the product)
Free Falls Withstanding Test	Height of fall (units within packing) 1 m	DIN EN 61131-2 DIN EN 60068-2-32
Vibration Resistance	10 Hz - 57 Hz: 0,0375 mm - continuous amplitude (0,075 mm peak amplitude) 57 Hz -150 Hz: 0.5 constant acceleration for continuous operation (1 g constant acceleration occasionally) 1 octave/minute, 10 frequency sweeps (sinusoidal), all 3 spatial axes	DIN EN 61131-2 DIN EN 60068-2-6
Shock Resistance	15 g occasionally, 11 ms, sinusoidal half-wave, 2 shocks in all three spatial axes	DIN EN 61131-2 DIN EN 60068-2-27
Class of Protection	When correctly installed in the rack: IP 20	DIN EN 60529
Mounting Position	Generally vertically	

Electrical Safety

Features	Value(s) (depending on the product)	Preferable reference standard(s) (depending on the product)
Protection class	III	DIN EN 61131-2
Dielectric Test Voltage	Functional ground is connected to chassis ground internally.	DIN EN 61131-2
Overvoltage Category	II	DIN EN 61131-2

EMC
Emitted interference

Features	Value(s) (depending on the product)	Preferable reference standard(s) (depending on the product)
Enclosure	Frequency band 30 -230 MHz, limit 30 dB (µV/m) at 30 m Frequency band 230 -1000 MHz, limit 37 dB (µV/m) at 30 m (class A)	DIN EN 50081-2 DIN EN 55011

EMC
Immunity to interference
- Housing -

Features	Value(s) (depending on the product)	Preferable reference standard(s) (depending on the product)
RF-Field, amplitude-modulated	Frequency band 26-1000 MHz Test field strength 10 V/m AM 80 % with 1 kHz Criterion A	DIN EN 61131-2 DIN EN 61000-6-2 DIN EN 61000-4-3
Electromagnetic RF-field, pulse-modulated	Frequency 900 +/-5 MHz Test field strength 10 V/m 50 % ON period Repetition rate 200 Hz criterion A	DIN EN 61000-6-2 DIN EN 61000-4-3
Magnetic Field with Mains Frequency	50 Hz 30 A/m	DIN EN 61000-6-2 DIN EN 61000-4-8
ESD	Discharge through air: Test peak voltage 8 kV Contact Discharge: Test peak voltage 4 kV Criterion A	DIN EN 61131-2 DIN EN 61000-6-2 DIN EN 61000-4-2

EMC
Immunity to interference
- Signal ports

Features	Value(s) (depending on the product)	Preferable reference standard(s) (depending on the product)
Asymmetric RF, amplitude-modulated	Frequency band 0.15 - 80 MHz Test voltage 10 V AM 80 % with 1 kHz Source impedance 150 ohms Criterion A	DIN EN 61000-6-2 DIN EN 61000-4-6
Burst mit schnellen Transienten (fast transients)	Test voltage 1 kV tr/tn 5/50 ns Repetition rate 5 kHz Criterion A	DIN EN 61000-6-2 DIN EN 61131-2 DIN EN 61000-4-4
Test with damped oscillation	Damped sinusoidaloscillation Frequency 1 Mhz Source impedance 200 ohms Repeat Factor 400/s Test voltage 1 kV	DIN EN 61131-2 DIN EN 61000-4-12

EMC
Immunity to interference
- Process, measuring and control lines

Features	Value(s) (depending on the product)	Preferable reference standard(s) (depending on the product)
Asymmetric RF, amplitude-modulated	Frequency band 0.15 - 80 MHz Test voltage 10 V AM 80 % with 1 kHz Source impedance 150 ohms Criterion A	DIN EN 61000-6-2 DIN EN 61000-4-6
Burst with fast transients	Test voltage 2 kV tr/tn 5/50 ns Repetition rate 5 kHz Criterion A	DIN EN 61000-6-2 DIN EN 61131-2 DIN EN 61000-4-4

EMC
Immunity to interference
- DC and AC power supply IOs

Features	Value(s) (depending on the product)	Preferable reference standard(s) (depending on the product)
Asymmetric RF, amplitude-modulated	Frequency band 0.15 - 80 MHz Test voltage 10 V AM 80 % with 1 kHz Source impedance 150 ohms Criterion A	DIN EN 61000-6-2 DIN EN 61000-4-6
Burst mit schnellen Transienten (fast transients)	Test voltage 2 kV tr/tn 5/50 ns Repetition rate 5 kHz Criterion A	DIN EN 61000-6-2 DIN EN 61131-2 DIN EN 61000-4-4
Test with damped oscillation	Damped sinusoidaloscillation Frequency 1 Mhz Source impedance 200 ohms Repeat Factor 400/s Test voltage 1 kV	DIN EN 61131-2 DIN EN 61000-4-12

EMC
Immunity to interference
- Functional Earth Connections

Features	Value(s) (depending on the product)	Preferable reference standard(s) (depending on the product)
RF, asymmetric	Frequency band 0.15 - 80 MHz Test voltage 10 V AM 80 % with 1 kHz Source impedance 150 ohms Criterion A	DIN EN 61000-6-2 DIN EN 61000-4-6
Bursts	Test voltage 1 kV tr/tn 5/50 ns Repetition rate 5 kHz Criterion A	DIN EN 61000-6-2 DIN EN 61000-4-4

5 JX6 Rack

Four rack types are available:
- 1 slot, CPU-module
- 2 slots, CPU-module and 1 peripheral module
- 4 slots, CPU-module and 3 peripheral modules
- 8 slots, CPU-module and 7 peripheral modules

5.1 Physical Dimensions

5.1.1 JX6-Basis-1 with One Slot

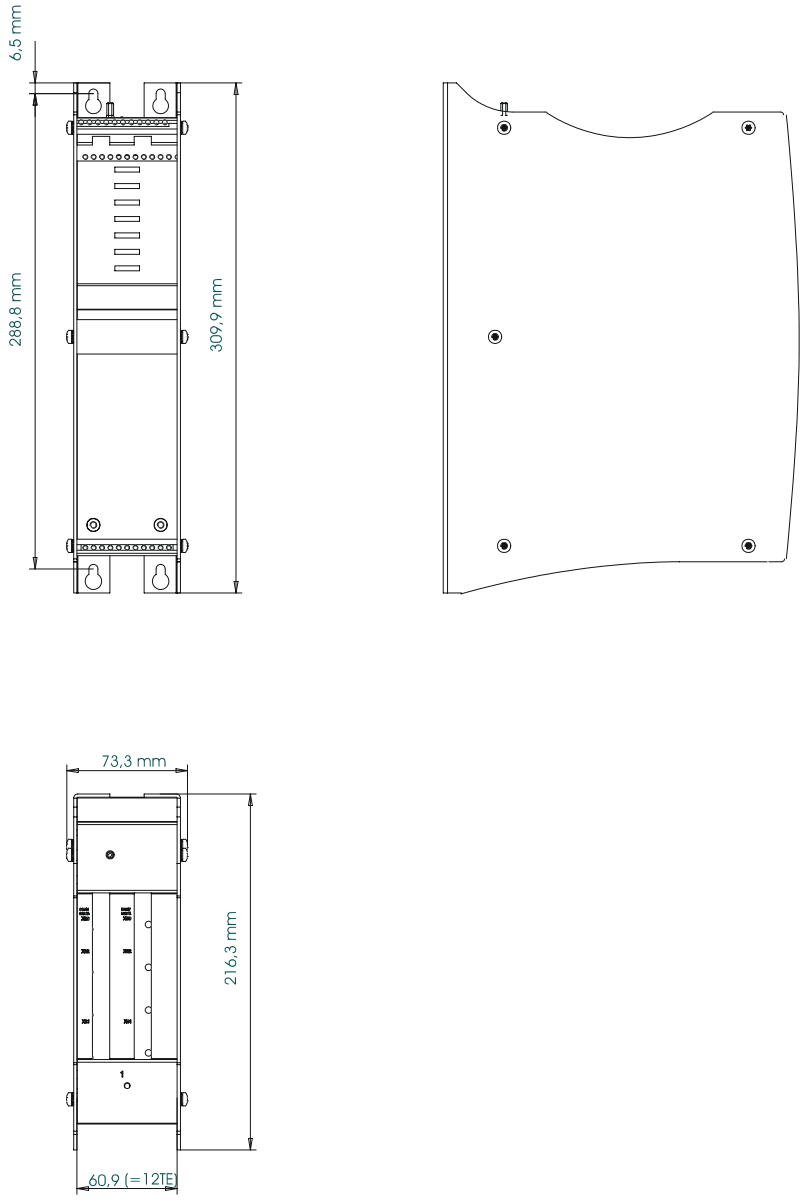


Fig. 5: JX6-Basis-1 Rack

5.1.2 JX6-Basis-2 with Two Slots

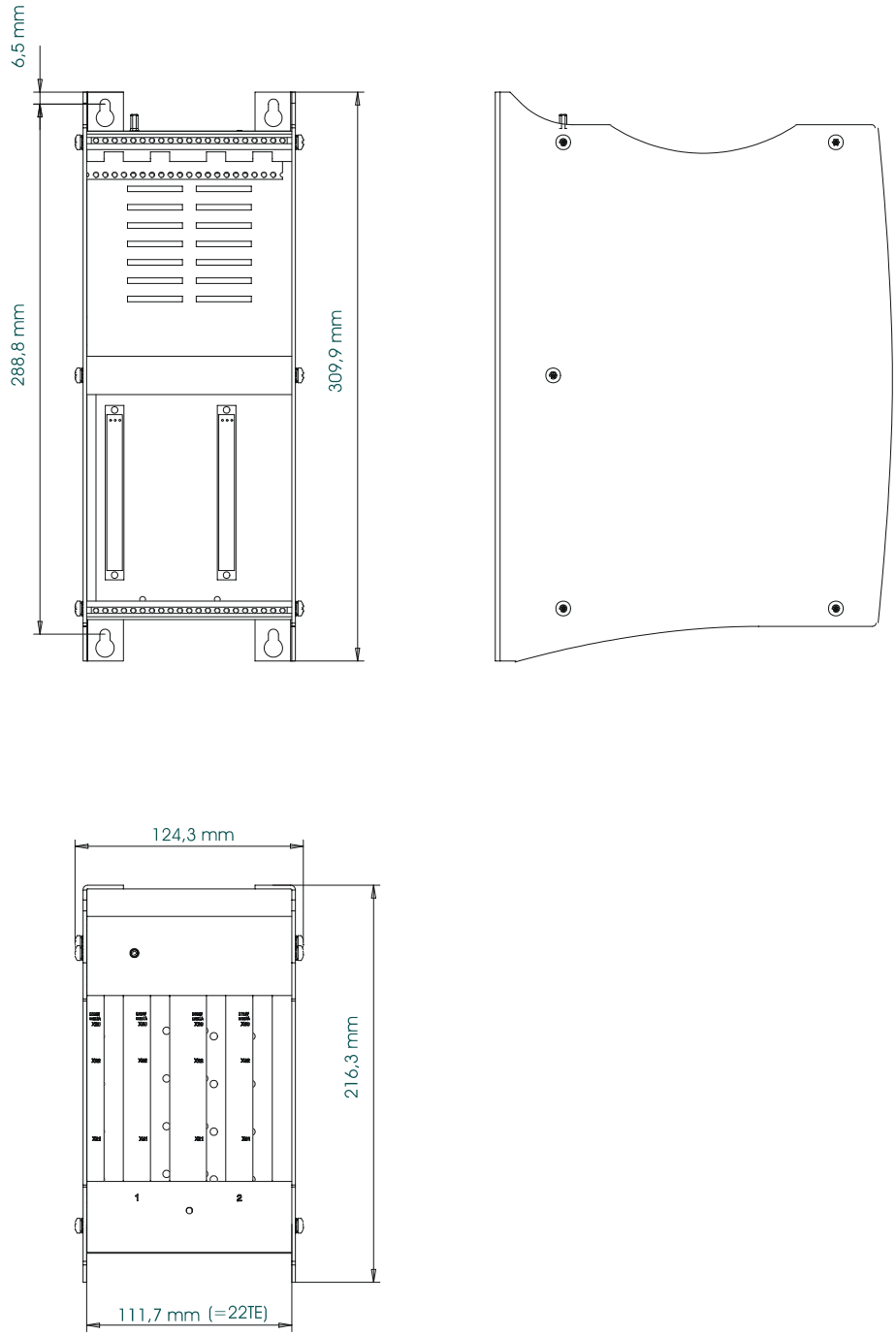


Fig. 6: JX6-Basis-2 Rack

5.1.3 JX6-Basis-4 with Four Slots

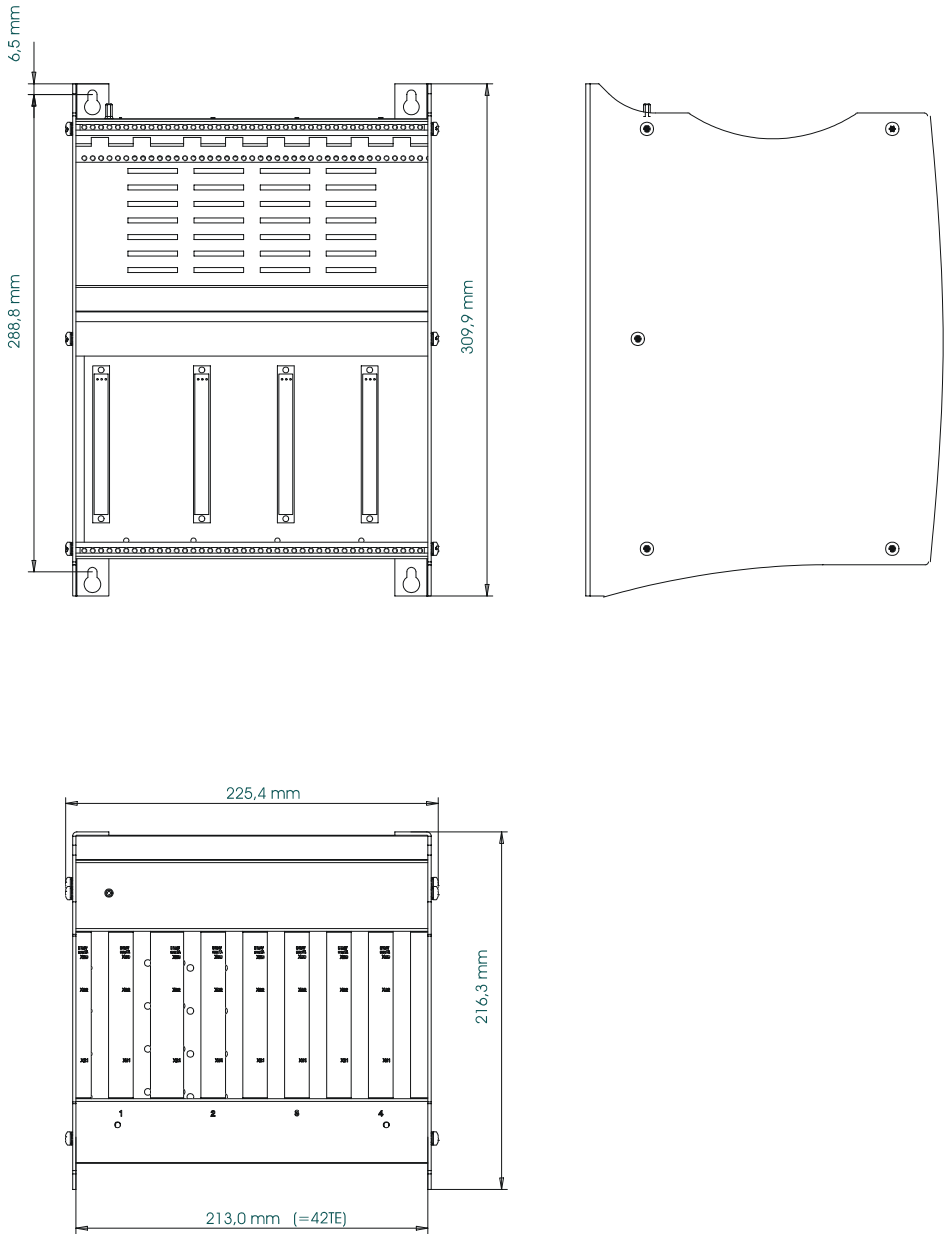


Fig. 7: JX6-Basis-4 Rack

5.1.4 JX6-Basis-8 with Eight Slots

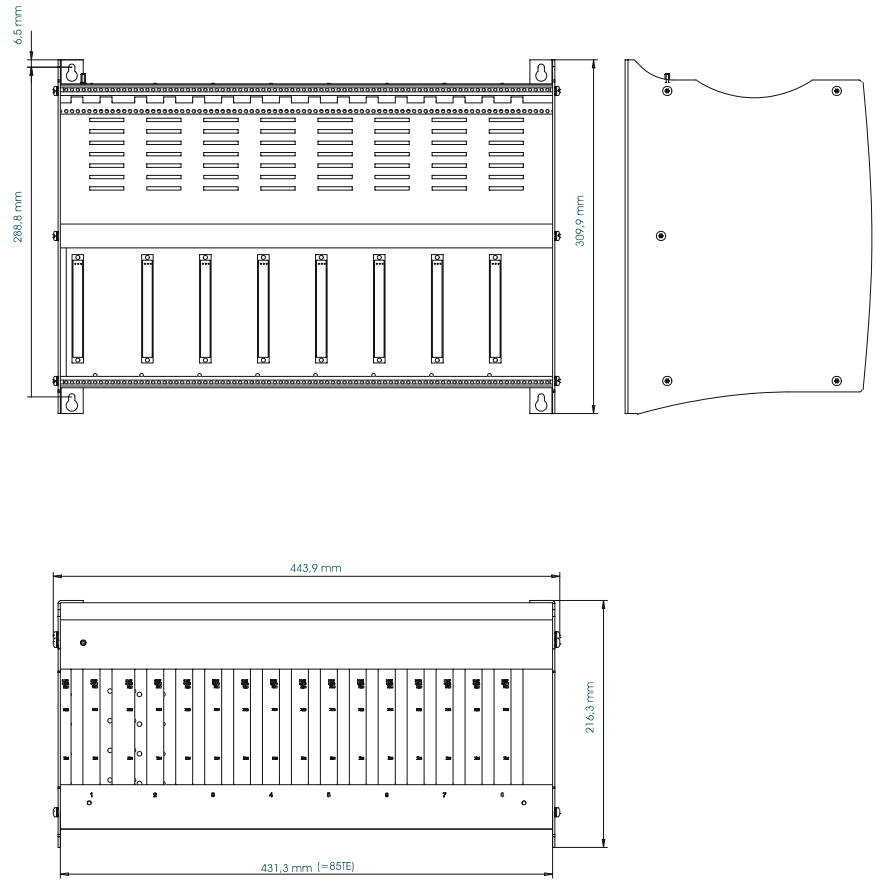


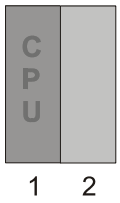
Fig. 8: JX6-Basis-8 Rack

5.2 Numbering of Slots

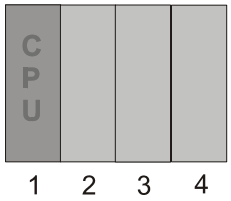
The slots are numbered in ascending order from left to right. Number 1 is assigned to the left slot and is reserved for the CPU-module. The other slots (2 through 8) can be equipped with peripheral modules (JX6-CON1 or JX6-IO64).

JX6-Basis-1:

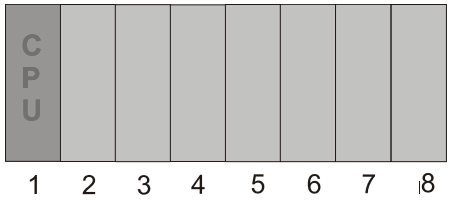
JX6-Basis-2:



JX6-Basis-4:



JX6-Basis-8:



6 Installation Guide

6.1 Deinstalling the JC-647 Module

Procedure

- Switch off the power supply for the controller.
- Remove the 2-pin terminal (1) of the power supply for the JC-647 module (see Fig. 9).
- Remove the two 8-pin terminals (2) of the digital inputs 1 through 16 on the JC-647 module (see Fig. 9).
- If applicable:
  Remove the 2-pin terminal block (3) of the external voltage supply of the outputs and the two 8-pin terminal blocks (3) of the digital outputs 1 to 16
  or
  the two 8-pin terminals (4) of the digital inputs 33 to 48 (see Fig. 9).

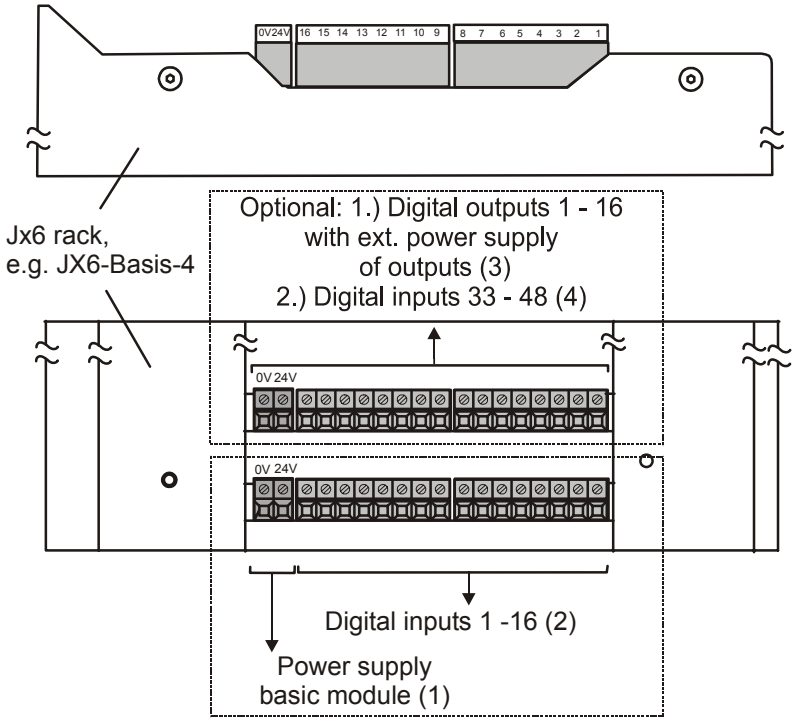

0V 24V
16 15 14 13 12 11 10 9
8 7 6 5 4 3 2 1
Jx6 rack,
e.g. JX6-Basis-4
Optional: 1.) Digital outputs 1 - 16
with ext. power supply
of outputs (3)
2.) Digital inputs 33 - 48 (4)
0V 24V
0V 24V
Digital inputs 1 -16 (2)
Power supply
basic module (1)

Fig. 9: Top and Side Views of the JX6 Rack

- Remove all Sub-D connectors (4) plugged into the JC-647 module. These connectors are located on the front side of the controller module.
- Remove all miniature DIN connectors (5) plugged into the JC-647 module. These connectors are located on the front side of the controller module.
- Use a screwdriver to loosen the four screws (6) connecting the JC-647 module with the JX6 rack (refer to Fig. 10).
- Pull the JC-647 module out of the JX6 rack using the two handles (7) (see Fig. 10).

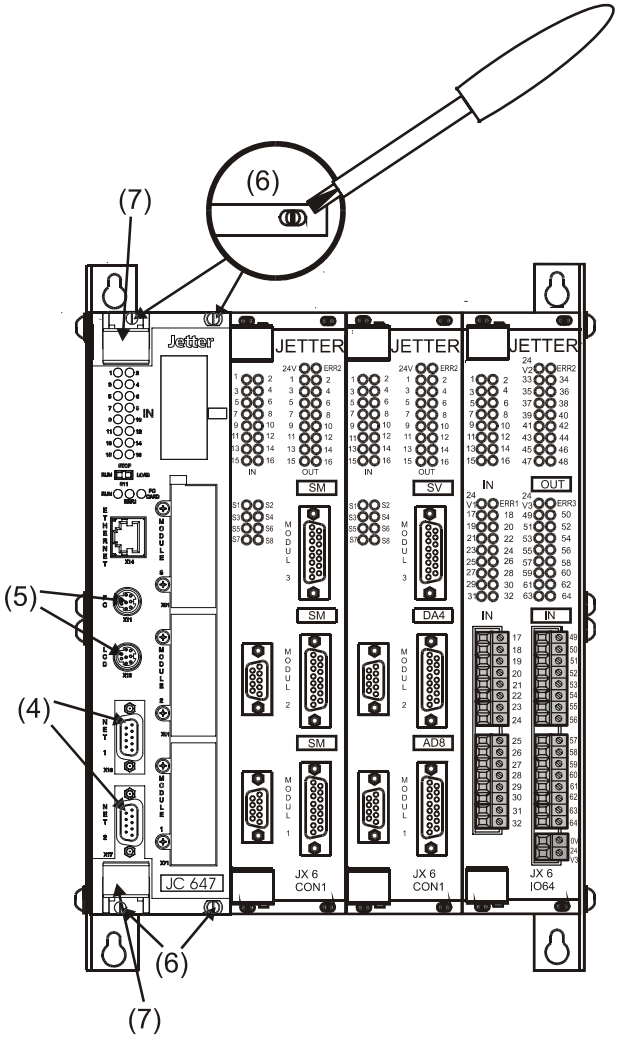


Fig. 10: Front View of the JX6 Rack with Modules

6.2 Installing the JC-647 Module

Procedure:

Install in reverse order of deinstallation.

Important!

Be sure to plug the green screw plug connectors into the correct receptacles. Connectors plugged into the wrong socket may cause malfunctions and even may destroy the control system.

Connectors are easily plugged into the wrong sockets by mistake especially if they are located close to each other.
To avoid this, Jetter AG provides a package of keying pins as standard with each control system. When properly installed, these keying pins prevent from plugging connectors incorrectly.

A suggestion for inserting the keying pins is described in chapter 6.3 "Using the keying pins" on page 42.

Be sure to supply the JC-647 module with voltage after its installation in the JX6 rack.
Otherwise, the control system will not be ready for operation.

6.3 Using the keying pins

A package of keying pins is provided with the control system. These keying pins are intended to prevent from incorrect plugging of screw terminals.

Important!

Screw terminals plugged into the wrong socket may result in malfunctions and even may destroy the control system.

The proper usage of keying pins is described below.

Appearance

The keying pin consists of two parts, For an illustration of these two parts see Fig. 11.

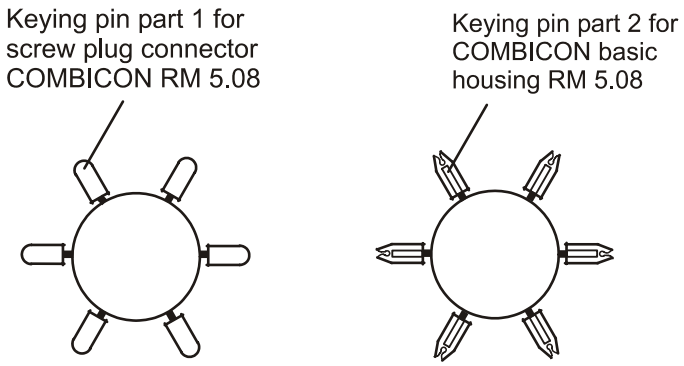


Fig. 11: Wheel of keying pins provided with the control system

Coding a plug connector

Fig. 12 gives an example of the keying procedure.

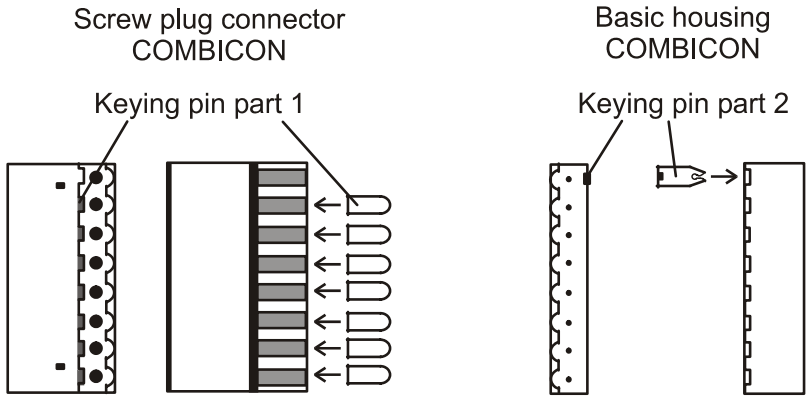


Fig. 12: Usage of keying pins

Suggestion for keying

Fig. 13 shows a suggestion for the keying of plug connectors.

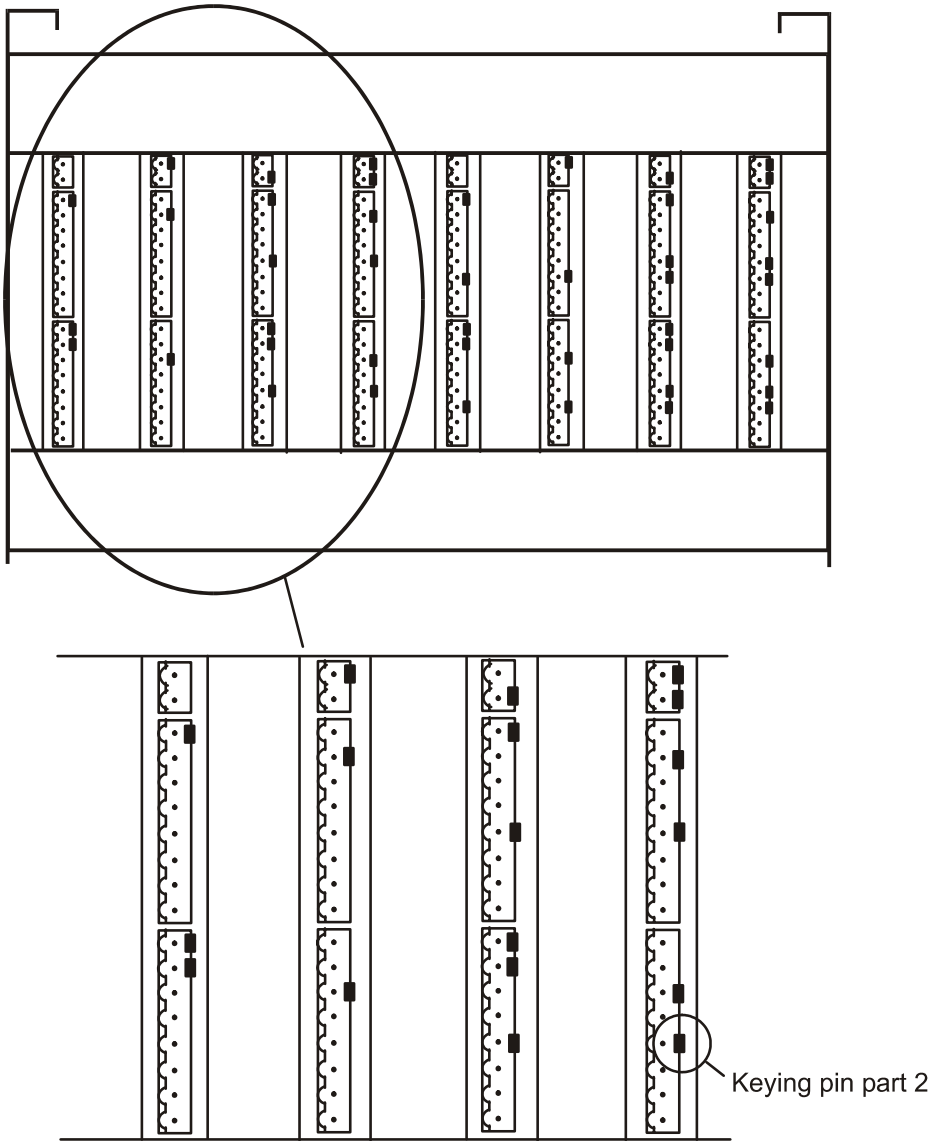


Fig. 13: Top view of the JXC-Basis-4 Rack with Keyed Connectors.

7 Submodules for Sockets # 1 through 3

7.1 Submodule types

Non-Intelligent Submodules

JX6-SV	1 incremental encoder, 1 analog output +/- 10 V
JX6-AD8	Analog-digital converter 8 voltage inputs or current inputs (single ended), or 4 voltage inputs or current inputs (differential)
JX6-DA4	Analog-digital converter 4 voltage or current outputs
JX6-Printer	1 printer activation module

For more information refer to the following chapters:
- Submodule JX6-SV - refer to chapter 7.6 "Submodule JX6-SV" on page 57;
- Submodule JX6-AD8 - refer to chapter 7.4 "JX6-AD8 Submodule" on page 50;
- Submodule JX6-DA4 - refer to chapter 7.5 "Submodule JX6-DA4" on page 54;

Intelligent Submodules

JX6-SB	1 CAN interface module for connection of modules to Jetter's system bus. This interface module serves for connecting JX2-I/O, JX-SIO, and third-party modules to the JetControl 647. It also enables networking of several controllers of the JetControl 647 and DELTA family in "Master/Master" mode.
JX6-SB-I	Besides the features of the JX6-SB module, the JX-SB-I module allows connection of JX2-Slave and JetMove modules to Jetter's system bus.
JX6-INT1	2 serial interfaces up to 115 KBaud. Special Protocols: Wipotec, Mettler, Dust, RemoteScan, HPGL, Multimaster mode
JX6-Profi	1 profi bus DP interface module
JX6-VCS	Valve control system
JX6-AS-INT1	Advanced serial interface

For more information on the JX6-SB(-I) module refer to chapter 19 "Submodules JX6-SB, and JX6-SB-I" on page 147.

Note!

Each submodule can be plugged on any socket (1, 2, or 3) (see Fig. 4).
Combination possibilities are almost unlimited, except for the JX6-DA4 where not more than two JX6-DA4 can be plugged.

Plugged on the JC-647, the functions of the submodules JX6-SV, JX6-AD8, JX6-DA4, JX6-INT, JX6-SB, and JX6-SB-I are fully supported.
The functions of the submodules JX6-Printer, JX6-Profi, JX6-VCS and JX6-AS-I can be activated if required.

The functionality of the above-mentioned submodules is described in a separate manual.

7.2 Submodule Type Registers

Note!

Modification:
The JC-647 detects the plugged submodule type and writes the type number of the plugged submodule into the submodule type register.
A submodule type register exists for each of the three sockets.

Type Number	Submodule Type
99	No module plugged
1	Reserved
2	Reserved
3	AD8_MODULE_TYPE
4	DA4_MODULE_TYPE
5	SV_MODULE_TYPE
6	PRN_MODULE_TYPE
72	INT_MODULE_TYPE
73	SB_MODULE_TYPE
74	AS-INTERFACE_MODULE_TYPE
75	PROFI_MODULE_TYPE
76	VCS_MODULE_TYPE

Register 61688: Submodule type of socket # 1	
Function	**Description**
Read	Setting the submodule type of socket # 1
Write	Illegal
Value range	1 to 6, 72 to 76, 99
Value after reset	99

Register 61689: Submodule type of socket # 2	
Function	**Description**
Read	Setting the submodule type of socket # 2
Write	Illegal
Value range	1 to 6, 72 to 76, 99
Value after reset	99

Register 61692: Submodule type of socket # 3	
Function	**Description**
Read	Setting the submodule type of socket # 3
Write	Illegal
Value range	1 to 6, 72 to 76, 99
Value after reset	99

7.3 Register Areas of Submodules

7.3.1 Socket 1

Non-intelligent submodule:	63300 .. 63399
Intelligent submodule:	111100 .. 111463

7.3.2 Socket 2

Non-intelligent submodule:	63400 .. 63499
Intelligent submodule:	112100 .. 112463

7.3.3 Socket 3

Non-intelligent submodule:	63500 .. 63599
Intelligent submodule:	113100 .. 113463

7.4 JX6-AD8 Submodule

The JX6-AD8 submodule is used for acquiring analog voltages and currents. The resolution of the AD converter is 16 bits.

Through a configuration, the following parameters can be specified:

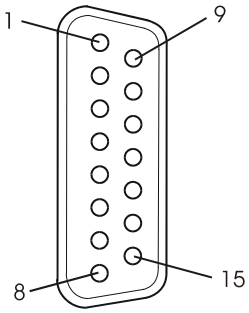
- Single-ended voltage channel (with reference to ground)
- Differential voltage channel
- Single-ended current channel (with reference to ground)
- Differential current channel
- Voltage range: -10 .. +10 V
- Current range: -20 .. +20 mA
- Current range: 4 .. 20 mA

The JX6-AD8 submodule may be plugged into sockets 1 through 3 of the JC-647 controller.

For detailed information on the JX6-AD8 submodule refer to the Operator's Manual of the JX6-AD8 submodule.

7.4.1 Description of Connections

Analog Voltage Channels

Pin Assignment - Female SUB-D connector, 15 pins				
View	**Pin**	**Signal**		**Comment**
		**Single-ended**	**Differential mode INxA -> INxB**	
	1	GND		Reference potential
	2	IN1	IN1 A	Analog input
	3	IN2	IN2 A	Analog input
	4	IN3	IN3 A	Analog input
	5	IN4	IN4 A	Analog input
	6	IN5	IN1 B	Analog input
	7	IN6	IN2 B	Analog input
	8	IN7	IN3 B	Analog input
	9	IN8	IN4 B	Analog input
	10	Not assigned		
	11	+15 V		Loadability: 5 mA
	12	-15 V		Loadability: 5 mA
	13	GND		Reference potential
	14	Not assigned		
	15	Not assigned		

Analog Current Channels

By inserting specific jumpers on the JX6-AD8 submodule, current channels can be allocated to the 15-pin female Sub-D connector (for more information refer to the JX6-AD8 Operator's Manual).

7.4.2 Overview of Registers

Addressing the submodule socket

Submodule socket # 1: y = 3

Submodule socket # 2: y = 4

Submodule socket # 3: y = 5

Overview of Registers

Reg. #	Description	
63y05	Instruction Register	1
63y51	Actual value of AD conversion - channel # 1	-32.768 .. +32.767
63y52	Actual value of AD conversion - channel # 2	-32.768 .. +32.767
63y53	Actual value of AD conversion - channel # 3	-32.768 .. +32.767
63y54	Actual value of AD conversion - channel # 4	-32.768 .. +32.767
63y55	Actual value of AD conversion - channel # 5	-32.768 .. +32.767
63y56	Actual value of AD conversion - channel # 6	-32.768 .. +32.767
63y57	Actual value of AD conversion - channel # 7	-32.768 .. +32.767
63y58	Actual value of AD conversion - channel # 8	-32.768 .. +32.767
63y59	Selecting the analog channels to be converted	1 .. 255 (bit-coded)
63y61	Configuration of analog channel # 1	3, 7, 8, 12, 17, 21
63y62	Configuration of analog channel # 2	3, 7, 8, 12, 17, 21
63y63	Configuration of analog channel # 3	3, 7, 8, 12, 17, 21
63y64	Configuration of analog channel # 4	3, 7, 8, 12, 17, 21
63y65	Configuration of analog channel # 5	3, 7, 8, 12, 17, 21
63y66	Configuration of analog channel # 6	3, 7, 8, 12, 17, 21
63y67	Configuration of analog channel # 7	3, 7, 8, 12, 17, 21
63y68	Configuration of analog channel # 8	3, 7, 8, 12, 17, 21

Possible Configurations

Input Configuration		
		Register Value
Single-ended	-20 .. +20 mA	3
	4 .. 20 mA	17
	-10 .. +10 V	8
Differential mode	-20 .. +20 mA	7
	4 .. 20 mA	21
	-10 .. +10 V	12

7.5 Submodule JX6-DA4

The JX6-DA4 submodule is used for converting digital data into analog voltages or currents. The resolution of the DA converter is 16 bits.

As outputs 4 single-ended channels are available. Each channel can be used as voltage or current output.

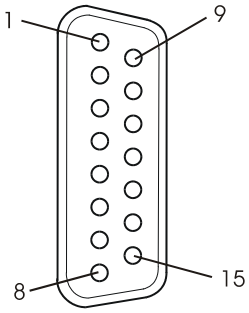
Voltage output range:	-10 V .. +10 V
Current output range:	0 mA ... 20 mA

Important!

- Do not pick off voltage **and** current simultaneously at the same DA channel.
- Do not plug more than two JX6-DA4 submodules on the sockets of the JC-647. Else, overload of the DC/DC converter will occur. The JC-647 is equipped with a 7.5 W DC/DC converter.

For detailed information on the JX6-DA4 submodule refer to the Operator's Manual of the JX6-DA4 submodule.

7.5.1 Description of Connections

Pin Assignment - Female SUB-D connector, 15 pins			
View	**Pin**	**Signal**	**Comment**
	1	GND	Reference potential
	2	Not assigned	
	3	IOUT4	Current output - Channel # 4
	4	IOUT3	Current output - Channel # 3
	5	IOUT2	Current output - Channel # 2
	6	IOUT1	Current output - Channel # 1
	7	GND	Reference potential
	8	GND	Reference potential
	9	GND	Reference potential
	10	GND	Reference potential
	11	VOUT4	Voltage output, channel # 4
	12	VOUT3	Voltage output, channel # 3
	13	VOUT2	Voltage output, channel # 2
	14	VOUT1	Voltage output, channel # 1
	15	Not assigned	

7.5.2 Overview of Registers

Addressing a submodule socket

Submodule socket # 1: y = 3

Submodule socket # 2: y = 4

Submodule socket # 3: y = 5

Overview of Registers

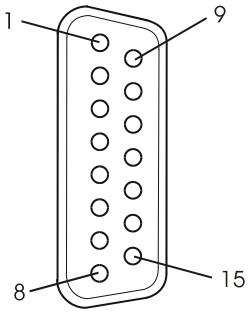
Reg. #	Description	
63y01	DAC output value - channel # 1	-32.768 .. +32.767
63y02	DAC output value - channel # 2	-32.768 .. +32.767
63y03	DAC output value - channel # 3	-32.768 .. +32.767
63y04	DAC output value - channel # 4	-32.768 .. +32.767

7.6 Submodule JX6-SV

The JX6-SV submodule can be used either as fast dual-channel counter or to control servo axes.
This chapter describes the dual-channel counter function. A description of the servo controller function is given in the Operator's Manual of the JX6-SV Controller.

The JX6-SV submodule may be plugged into sockets 1 through 3 of the JC-647 controller.

7.6.1 Description of Connections

Pin Assignment - Female SUB-D connector, 15 pins			
View	**Pin**	**Signal**	**Comment**
	1	GND	
	2	K0 (+)	Zero pulse
	3	K0 (-)	Zero pulse
	4	K1 (+)	Counter channel # 1
	5	K1 (-)	Counter channel # 1
	6	K2 (+)	Counter channel # 2
	7	K2 (-)	Counter channel # 2
	8	Reserved	
	9	Reserved	
	10	DC 5 V (max. 50 mA)	Voltage supply for incremental encoder
	11	Unassigned	
	12	Unassigned	
	13	DA output +/- 10 V	
	14	Relay contact 1	N/O
	15	Relay contact 2	floating

7.6.2 Overview of Registers

Addressing a submodule socket

Submodule socket # 1: y = 3

Submodule socket # 2: y = 4

Submodule socket # 3: y = 5

Overview of Registers

Reg. #	Description	
63y01	Instruction Register	
63y09	Actual position (4-fold evaluation)	
63y33	Relay	0: OFF 1: ON
63y60	DA output	+/-32,767 (corresponds to +/-10 V)

8 PC-Card

The PC-card is optional. Either a PC-card or a third socket is possible.

8.1 Hardware

- The JC-647 is equipped with a programmable decoder chip (CPLD) with the designation D_ADR2 to address the PC-card interface.
- Memory cards are supported according to PCMCIA V2.01 type I (thickness 3.3 mm).
- Cards with max. 8 MBytes can be addressed, which results in a maximum number of 2.097.152 registers.
- Static RAM memory cards are available with the following memory sizes:

Memory Size	Quantity of Registers	Register Numbers
1 MBytes	262144	393216 .. 655259
2 MBytes	524288	393216 .. 917503
4 MBytes	1048576	393216 .. 1441791
8 MBytes	2097152	393216 .. 2490367

8.2 Software

- The PC-card is used as expansion memory for the user registers. For this purpose, the common memory area of the card is inserted into the memory area of the processor of the JC-647 and is directly accessed by the processor (no disadvantage over internal registers with respect to speed). Any formatting of the card carried out by a PC operating system is not taken into account. Four bytes are assigned to each register.
- Only cards can be used that do not require a special read or write procedure, such as static RAM (usually battery-backed).
- The numbering of registers on the PC-card directly follows that of the internal extension registers (JC-647) (starting from register number 393216).
- DELTA hardware and software allow hot plugging, i.e. the PC card can be plugged and unplugged during operation without damaging the application. However, the card should not be removed when the user program is accessing registers of the card.
- The yellow LED (PC-card) on the front panel is lit when the PC-card is plugged and activated.
- By means of register 61445 can be determined whether a PC-card is plugged correctly, whether the operating system of the JC-647 has detected the card and whether card initialization is completed. The registers on the PC card can only be used when register 61445 contains the value 3. This is also the case for the three registers described in the following which only contain valid values if register 61445 = 3.

- Register 61446 indicates various status messages from the PC-card. This register is bit-coded:

Bit 7:	Always 0
Bit 6:	Always 0
Bit 5: **Bit 4:**	Battery status: 11 = ok 10 = Replace battery; data ok 01 = Battery empty; data not ok 00 = Battery empty; data not ok
Bit 3:	1 = card is write-protected
Bit 2:	1 = card is ready
Bit 1: **Bit 0:**	00 = card is inserted correctly 01 = card is inserted incorrectly 10 = card is inserted incorrectly 11 = card is inserted incorrectly

- If the card is active with CIS (CIS = Card Information Structure) according to PCMCIA, then register 61690 contains the device ID from tuple 01 device info 1. The device ID contains coded information about the device type (RAM, EPROM, Flash, etc.), handling of the write protect switch, and access speed.
  For cards before PCMCIA V2.1, CIS need not necessarily be used.
- If the card is active with CIS according to PCMCIA, then register 61691 contains the device size byte from tuple 01 device info 1 from which the memory size of the card in the common memory range can be determined.
  For cards before PCMCIA V2.1, CIS need not necessarily be used.

9 RUN - STOP - LOAD Switch

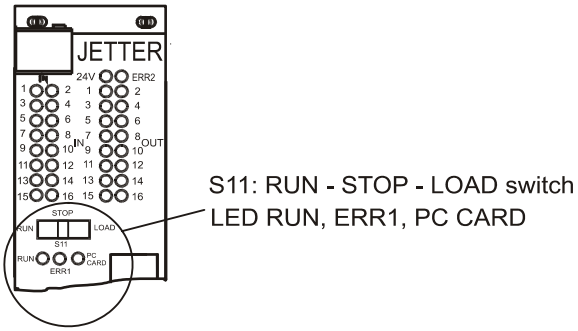


Fig. 14: RUN - STOP - LOAD Switch of the JC-647 Module

Switch	
Position	**Meaning**
RUN	Application program is running when the controller is switched on.
STOP	Application program is not running when the controller is switched on.
LOAD	Application program is stopped Operating system update is possible

Note!

The switch is scanned when the controller is switched on. Actuating the switch during operation of the controller is ignored.

LEDs		
Designation	**Colour**	**Function**
RUN	green	**ON:** Operating system and application program running **Flashing:** Application program not running.
ERR1	red	**ON:** Operating system error: error code in reg. 61477
PC CARD	yellow	**ON:** PC-card is plugged and activated

10 Power Supply

10.1 Requirements

Power Supply Unit Requirements	
Voltage Range	DC 20 .. 30 V Residual ripple<5 % filtered
Power consumption	Depending on the number of attached submodules: approx. 10 .. 15 W

Caution

Important!

- The maximum supply voltage must not exceed DC 30 V since a higher supply voltage may cause damages to the JC-647 module.
- If the JC-647 module is not supplied with sufficient power (under-voltage), malfunctions may occur.

Note!

Protection against polarity reversal:
The JC-647 module is protected against polarity reversal.
For testing the module, a voltage of 24 volts with reversed polarity was applied for 10 s.

10.2 Description of Connections

Terminal Specifications

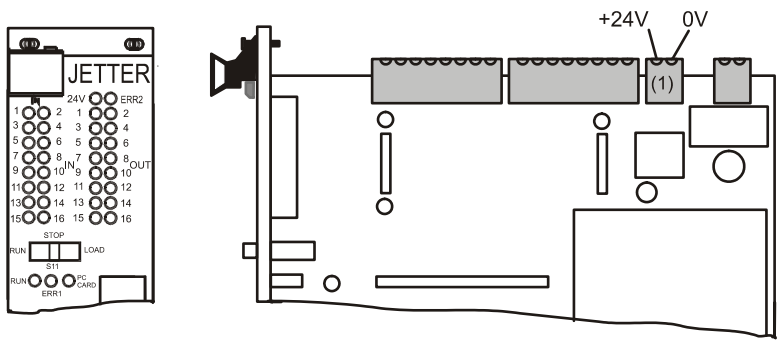
- 2-pole terminal block COMBICON RM 5.08 (for printed circuit boards)
- Cable cross-sectional area: 0.25 -2.5 mm^2
- Bladed screw-driver: 0.6 x 3.5 x 100 mm

Connecting Cable Specifications

- Not needed

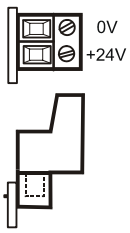
Cable Shielding

- Not needed



(1): Power supply terminal

Fig. 15: Power Supply Terminal of JC-647

Pin Assignment of 2-pin Plug for Terminal Blocks			
View	**Pin**	**Signal**	**Comment**
	+24 V	DC 20 .. 30 V	
	0 V	GND	

11 Digital Inputs

11.1 Technical Data

Digital inputs form an integral part of the JC-647.

Functional Data	
Number of inputs	16 digital inputs
Rated voltage	DC 24 V

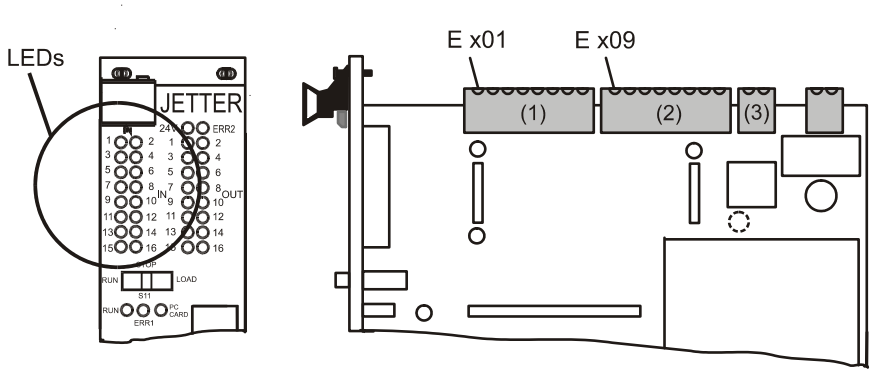
Electrical Data	
Voltage Range	DC 15 .. 27 V
Signal voltage ON	min. 15 V
Signal voltage OFF	max. 10 V
Input current	approx. 8 mA
Input resistance	3.0 kΩ
Input delay time	approx. 500 µs (with state change from 0 to 1, and 1 to 0)
Electrical isolation	None

Caution

Important!

If a line with reverse polarity is connected to a digital input, the input will be destroyed.

11.2 Meaning of LEDs



(1): Terminal for digital inputs 1 through 8
(2): Terminal for digital inputs 9 through 16

Fig. 16: Digital Input LEDs of the JC-647

LEDs of Digital Inputs		
Designation	**Colour**	**Function**
IN 1 .. 16	yellow	Digital input 1 through 16 **ON:** Signal voltage ON **OFF:** Signal voltage OFF

11.3 Description of Connections

Terminal Specifications

- 8-pole terminal block COMBICON RM 5.08 (for printed circuit boards)
- Diameter of the cable apt for connecting: 0.25 -2.5 mm^2
- Bladed screw-driver: 0.6 x 3.5 x 100 mm

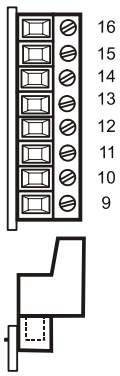
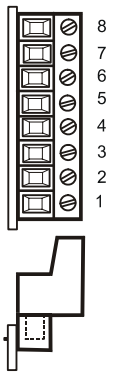
Connecting Cable Specifications

- Not needed

Cable Shielding

- Not needed

Pin Assignment of 8-pin Plug for Terminal Blocks

View	Pin	Signal	Comment
	16	Digital input # 16	
	15	Digital input # 15	
	14	Digital input # 14	
	13	Digital input # 13	
	12	Digital input # 12	
	11	Digital input # 11	
	10	Digital input # 10	
	9	Digital input # 9	
	8	Digital input # 8	
	7	Digital input # 7	
	6	Digital input # 6	
	5	Digital input # 5	
	4	Digital input # 4	
	3	Digital input # 3	
	2	Digital input # 2	
	1	Digital input # 1	

See Fig. 17 for the assignment of digital inputs of the JC-647 module. Reference point is the 0 V terminal to which the 0 V signal is connected-up.

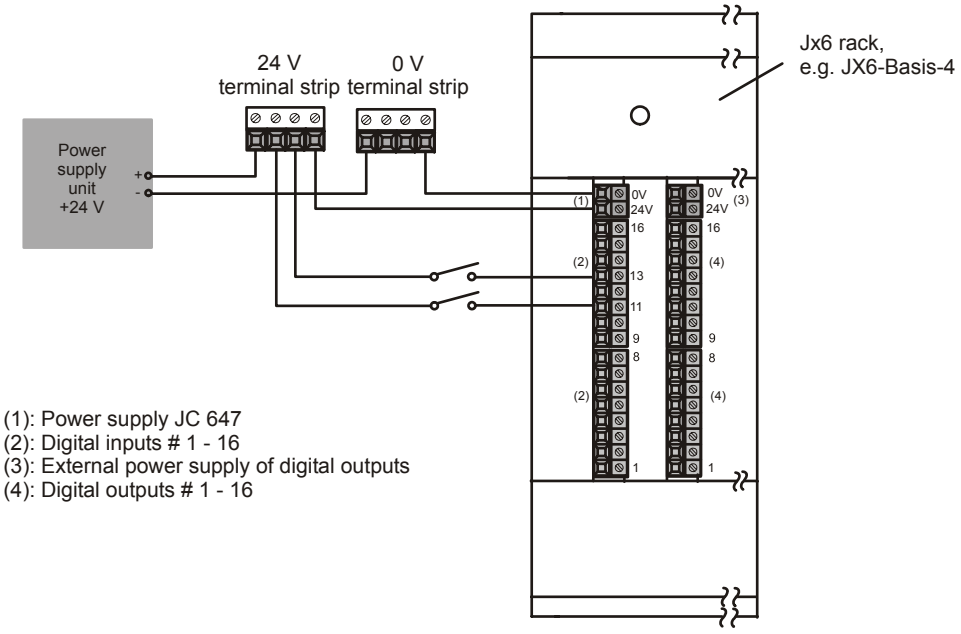


Fig. 17: External circuit of the digital inputs 11 and 13

11.4 Numbering System of Digital Inputs

Numbering System of Digital Inputs	
Digital Input JC-647	**Address Number**
Input # 1	100 + 1
Input # 2	100 + 2
...	...
Input # 16	100 + 16

12 Submodule JX6-I16

Optionally, the JC-647 module can be equipped with the submodule JX6-I16, which expands the function by 16 digital inputs.

12.1 Technical Data

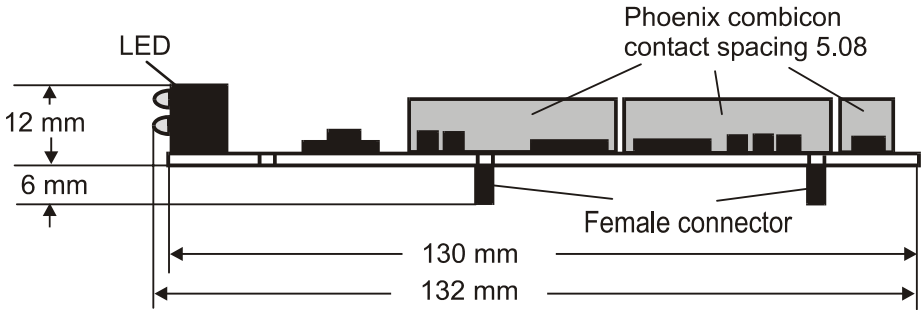


Fig. 18: Side View, Submodule JX6-I16

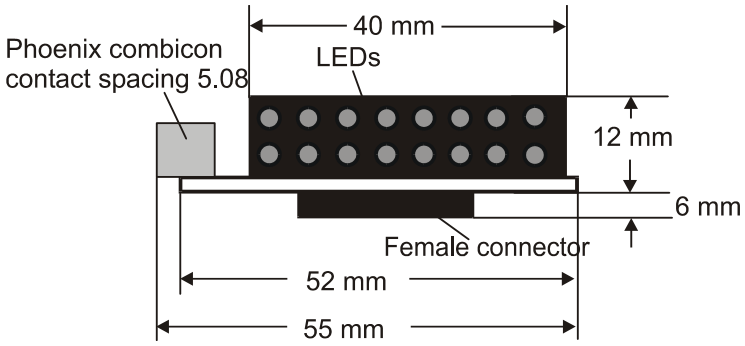


Fig. 19: Front View, Submodule JX6-I16

Type	
Dimensions (H x W x D in mm)	12.0 x 52.0 x 130.0
Installation	To be plugged on the JC-647 (see Fig. 4)

Functional Data	
Number of outputs	16 digital inputs
Rated voltage	DC 24 V

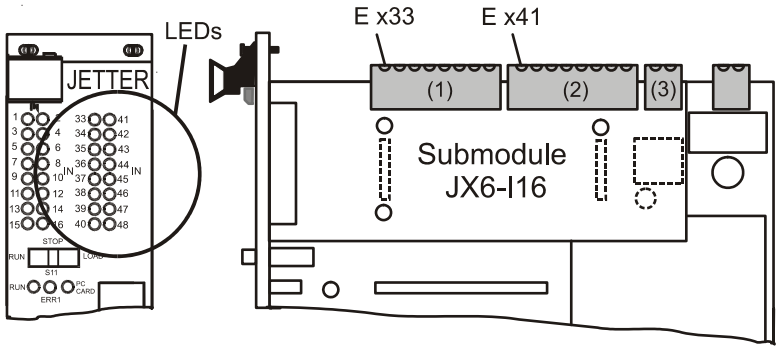
Electrical Data	
Voltage range	DC 15 ... 27 V
Signal voltage ON	min. 15 V
Signal voltage OFF	max. 10 V
Input current	approx. 8 mA
Input resistance	3,0 kΩ
Input delay time	approx. 3 ms
Electrical isolation	None

Caution

Important!

If a line with reverse polarity is connected to a digital input, the input will be destroyed.

12.2 Meaning of LEDs



(1): Terminal for digital inputs 33 to 40
(2): Terminal for digital inputs 41 to 48
(3): No function

Fig. 20: LEDs of the submodule JX6-I16

LEDs of the submodule JX6-I16		
Designation	Colour	Function
IN 33 ... 48	Yellow	Digital input 33 through 48 **ON:** Signal voltage ON **OFF:** Signal voltage OFF

12.3 Description of Connections

Terminal Specifications

- 8-pole terminal block COMBICON RM 5.08 (for printed circuit boards)
- Diameter of the cable apt for connecting: 0.25 -2.5 mm^2
- Bladed screw-driver: 0.6 x 3.5 x 100 mm

Connecting Cable Specifications

- Not needed

Cable Shielding

- Not needed

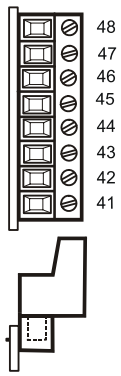
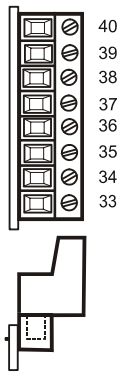
Pin Assignment of 8-pin Plug for Terminal Blocks			
View	**Pin**	**Signal**	**Comment**
	48	Digital input # 48	
	47	Digital input # 47	
	46	Digital input # 46	
	45	Digital input # 45	
	44	Digital input # 44	
	43	Digital input # 43	
	42	Digital input # 42	
	41	Digital input # 41	
	40	Digital input # 40	
	39	Digital input # 39	
	38	Digital input # 38	
	37	Digital input # 37	
	36	Digital input # 36	
	35	Digital input # 35	
	34	Digital input # 34	
	33	Digital input # 33	

Fig. 21 shows the assignment of digital inputs of the submodule JX6-I16. Reference point is the 0 V terminal to which the 0 V signal is connected-up.

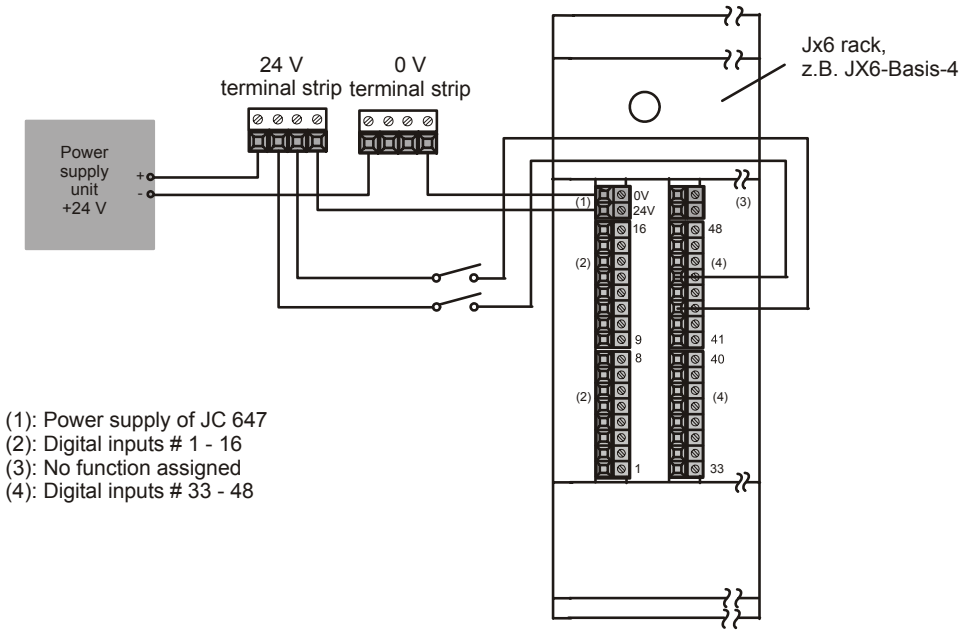


Fig. 21: External circuit of the digital inputs 43 and 45

12.4 Numbering System of Digital Inputs

Numbering System of Digital Inputs	
Digital Input JX6-I16	**Address Number**
Input # 33	100 + 33
Input # 34	100 + 34
...	...
Input # 47	100 + 47
Input # 48	100 + 48

13 Submodule JX6-O16

Optionally, the JC-647 module can be equipped with the submodule JX6-O16, which expands the function by 16 digital outputs.

13.1 Technical Data

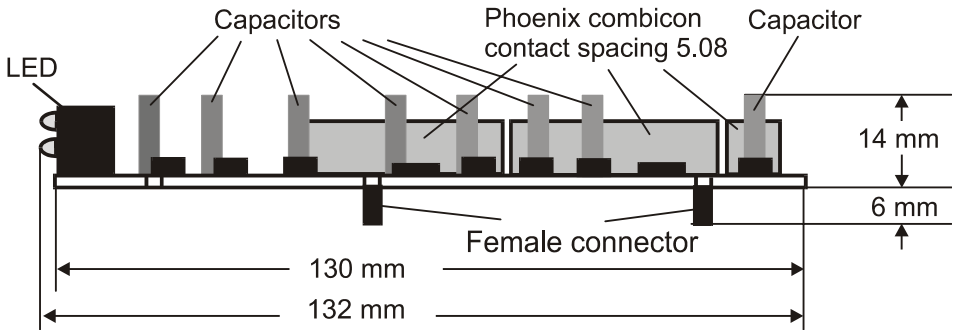


Fig. 22: Side View of the JX6-O16 Submodule

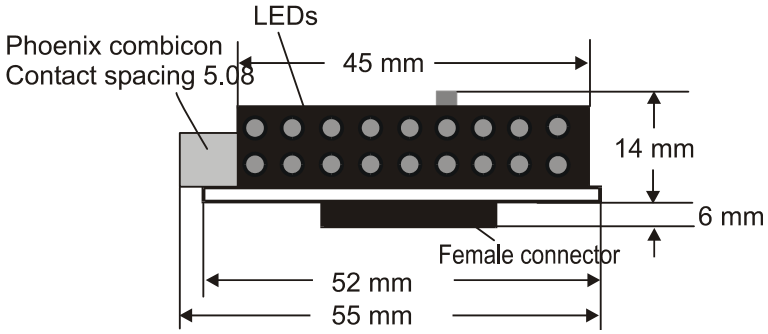


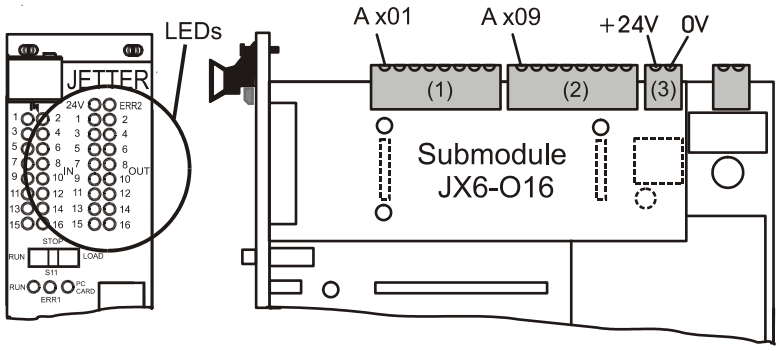
Fig. 23: Front View of the JX6-O16 Submodule

Type	
Dimensions (H x W x D in mm)	14.0 x 52.0 x 130.0
Installation	To be plugged on the JC-647 (see Fig. 4)
Type of outputs	Transistor, pnp

Functional Data	
Number of outputs	16 digital outputs
Rated voltage	DC 24 V
Type of outputs	Transistor, pnp

Electrical Data	
External power supply Voltage Range	Required DC 20 .. 30 V
Signal voltage ON	Typ. V_{supply} - 0,5 V
Signal voltage OFF	Typ. 0.8 V
Maximum load current	0,5 A / output
Electrical isolation	None
Protective circuit	Overload, overvoltage, overtemperature indicated by the red LED ERR2
Protection against inductive loads	Yes

13.2 Meaning of LEDs



(1): Terminal for digital outputs 1 to 8
(2): Terminal for digital outputs 9 to 16
(3): External power supply of digital outputs

Fig. 24: LEDs of the JX6-O16 Submodule

LEDs of the Digital Outputs		
Designation	**Colour**	**Function**
OUT 1 .. 16	yellow	Digital output 1 through 16 **ON:** Signal voltage ON **OFF:** Signal voltage OFF
ERR2	red	**ON:** $I_{max.}$ less than 10 mA Overload, overtemperature, cable breakage of one or more outputs
24V	green	**ON:** External voltage supply of the digital outputs is provided

13.3 Description of Connections

Terminal Specifications

- 8-pole terminal block COMBICON RM 5.08 (for printed circuit boards)
- Cable cross-sectional area: 0.25 -2.5 mm^2
- Bladed screw-driver: 0.6 x 3.5 x 100 mm

Connecting Cable Specifications

- Not needed

Cable Shielding

- Not needed

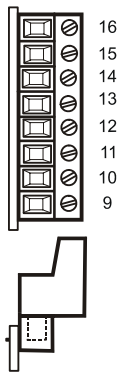
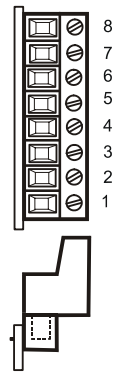
Pin Assignment of 8-pin Plug for Terminal Blocks			
View	**Pin**	**Signal**	**Comment**
	16	Digital Output # 16	
	15	Digital Output # 15	
	14	Digital Output # 14	
	13	Digital Output # 13	
	12	Digital Output # 12	
	11	Digital Output # 11	
	10	Digital Output # 10	
	9	Digital Output # 9	
	8	Digital Output # 8	
	7	Digital Output # 7	
	6	Digital Output # 6	
	5	Digital Output # 5	
	4	Digital Output # 4	
	3	Digital Output # 3	
	2	Digital Output # 2	
	1	Digital Output # 1	

Fig. 25 shows the connection of the digital outputs of the JX6-O16. The 0 V terminal strip to which the 0 V signal is applied is located in the switch cabinet.

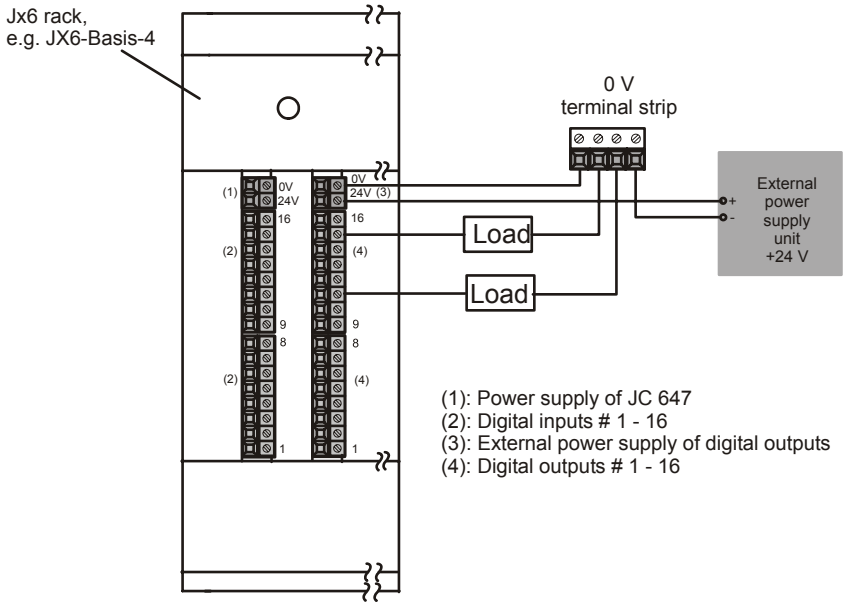


Fig. 25: External circuit of the digital outputs 11 and 15

13.4 Numbering system of digital outputs

Numbering system of digital outputs	
Digital output JX6-O16	**Address Number**
Output 1	100 + 1
Output 2	100 + 2
...	...
Output 16	100 + 16

14 Counters

14.1 Overview of Functions

The JC-647 is equipped with four counters. The digital inputs 101 through 104 serve as inputs for counters 1 through 4. After activation of the counters, the counting pulse present at inputs 101 through 104 go to the counter. Inputs 101 through 104 can in any case be used as digital inputs of the PLC.

There is a shared register for the counters from which the counter contents can be read. The respective counter must have been selected before the counter contents is read.

The sampling interval for each counter is 1 kHz. Thus, pulses up to a frequency of approx. 500 Hz can be counted.

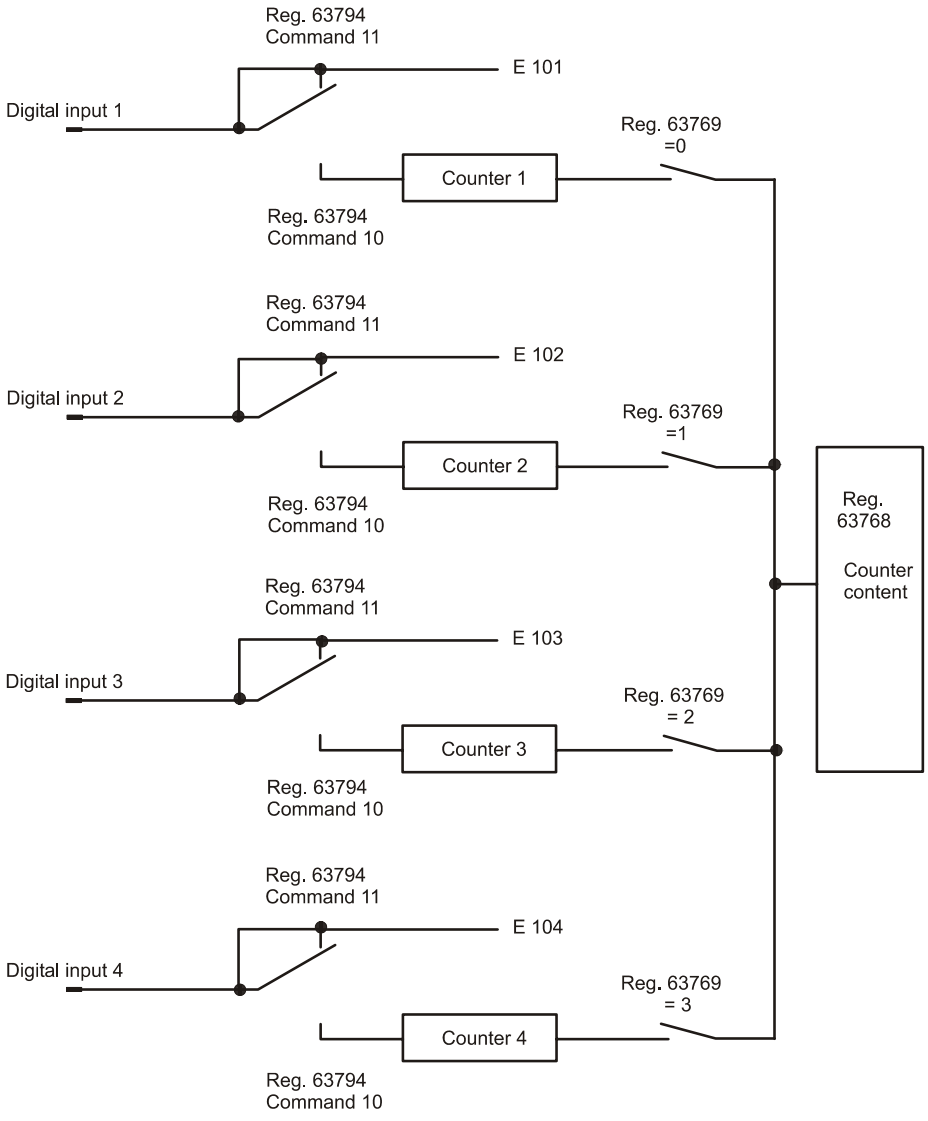


Fig. 26: Overview of Counters

14.2 Meaning of Registers

Register 63794: Instruction Register	
Function	**Description**
Read	Last command
Write	Command to activate and deactivate the four counters
Value range	10, 11
Value after reset	0

Commands:

10 **Activating the four counters**

Digital input 101 is connected with counter 1.
Digital input 102 is connected with counter 2.
Digital input 103 is connected with counter 3.
Digital input 104 is connected with counter 4.

11 **Deactivating the four counters**

Digital inputs 101 through 104 are digital inputs of the PLC.

Register 63769: Counter Selection Register	
Function	**Description**
Read	Selected counter
Write	Select counter
Value range	0 through 3
Value after reset	0

Setting	Select counter
0	Counter 1 selected
1	Counter 2 selected
2	Counter 3 selected
3	Counter 4 selected

Register 63768: Counter content	
Function	**Description**
Read	Current counter contents of the selected counter
Write	-
Value range	0 through 65,535
Value after reset	0

Example:

```
; activate counter
; select counter 3
; read counter contents of counter 3
   THEN
     REGISTER_LOAD [63794, 10]
     REGISTER_LOAD [63769, 2]
     REGISTER_LOAD [rHelp, @(63768)]
```

15 Serial Interfaces

The JC-647 is equipped with four ports for the various serial interfaces.

JC-647 Serial Interfaces		
Interface	**Function**	**Specs**
8-pin miniature DIN port PC	• Programming • Visualizing	RS-232 RS-232
8-pin miniature DIN port LCD	• HMIs • Visualizing	RS-422 RS-232
9 pin SUB-D port NET1	• JETWay-H, -R	RS-485
9 pin SUB-D port NET2	• JETWay-H, -R • user-programmable	RS-485 RS-422 RS-232

Serial interfaces can be added by means of submodules, e.g. JX6-INT. These submodules can be plugged into the three sockets on the JC-647.

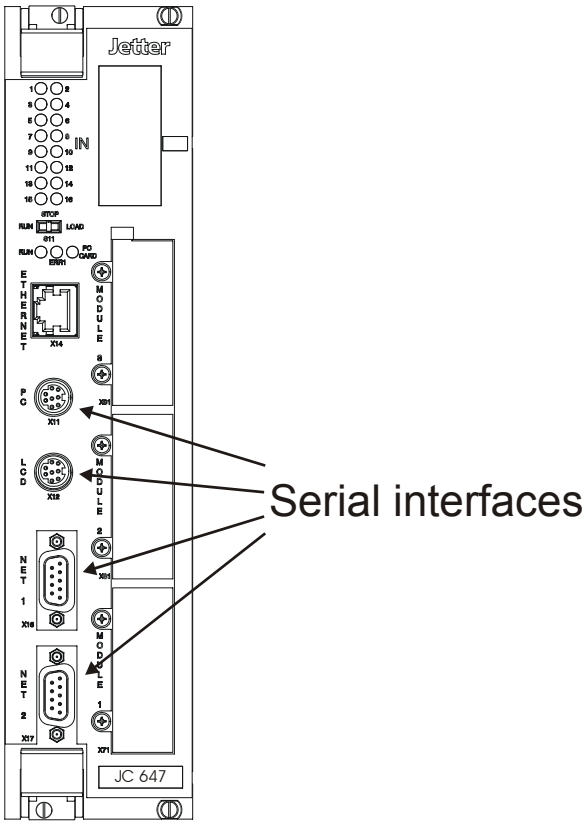
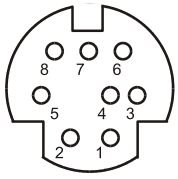


Fig. 27: Serial Interfaces of the JC-647 module

15.1 Programming Interface PC

This interface serves for communication between the JC-647 and the PC on which the JetSym programming interface is installed. The connection is realized by means of a programming cable. The interface is an RS-232 serial interface.

Pin Assignment 8-Pin Miniature DIN Jack PC			
View	**Pin**	**Signal**	**Comment**
	1	-	
	2	GND	Reference potential
	3	-	
	4	RXD	RS-232
	5	-	
	6	DC +24 V	Power Supply for HMI
	7	-	
	8	TxD	RS-232

15.1.1 Description of Connections

Specification of Connectors

On the controller side
- 8-pin miniature DIN male connector
  Type: KMDLA - 8P
  By: KYCON
- Diameter of the cable apt for connecting: 0.128 -0.051 mm^2

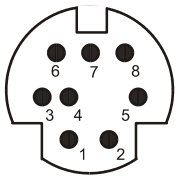
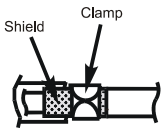
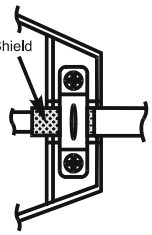
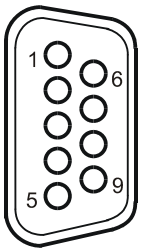
On the PC side
- 9-pin female SUB-D connector in metallized housing
- Diameter of the cable apt for connecting: 0.25 -0.128 mm^2

Connecting Cable Specifications

- Number of cores: 3
- Core cross-sectional area: 0.14 mm^2
- Max. cable length: 15 m

Cable Shielding

- Complete shielding, no paired shielding
- The shield must be connected to the connector housings on both ends of the cable with the greatest possible surface area (shield twisted, clamped under the strain relief clamps, and wrapped with copper foil).

Programming Cable JN-PK			
Controller	**Shield**		**PC**
			
	Connect shield with the greatest possible surface area! Use metallized housing only!		
Pin	**Signal**		**Pin**
8	TxD	RXD	2
4	RXD	TxD	3
2	GND		5
On the PC side (COM1, COM2), pins 7 and 8, as well as pins 1, 4 and 6 must be short-circuited.			

Note!

It is recommended to buy the cables from Jetter AG.

15.2 JetSym Settings

The interface type and the baud rate have to be set in JetSym.
This is carried out in the menu "Tools/Options/Controller":

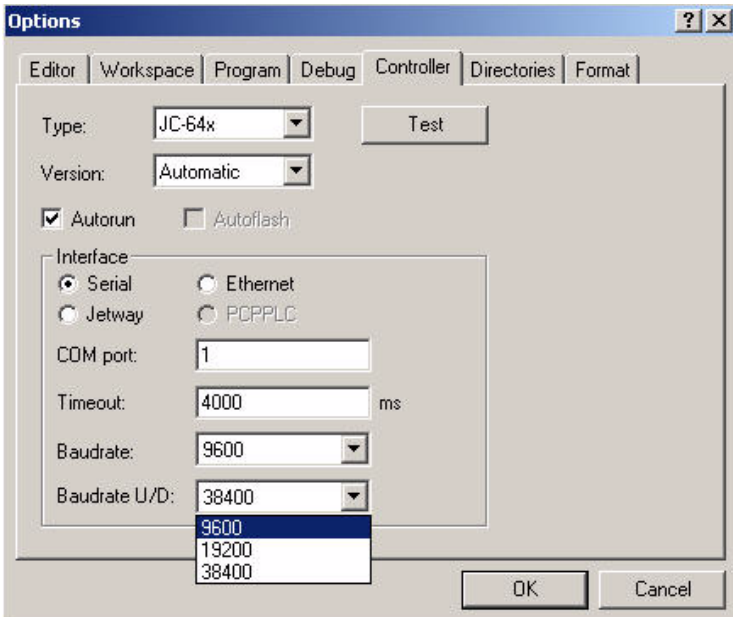
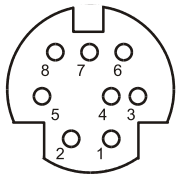


Fig. 28: Setting the Baud Rate for the Serial Interface

A baud rate of 9600, 19200 and 38400 Baud can be selected. This menu also allows to set the baud rate for uploading/downloading programs (Baud rate U/D).

15.3 HMI LCD Interface

The LCD interface serves for communication between user interface (HMI) and controller.

Pin Assignment of 8-Pin Miniature DIN Jack			
View	**Pin**	**Signal**	**Comment**
	1	RDA	RS-422
	2	GND	Reference potential
	3	RDB	RS-422
	4	RXD	RS-232
	5	SDB	RS-422
	6	DC +24 V	Power Supply for HMI
	7	SDA	RS-422
	8	TxD	RS-232

15.3.1 HMI Cable JN-DK-Xm

Specification of Connectors

On the controller side
- 8-pin miniature DIN male connector
  Type: KMDLA - 8P
  By: KYCON
- Diameter of the cable apt for connecting: 0.128 -0.051 mm^2

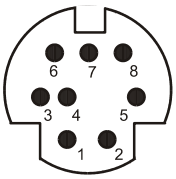
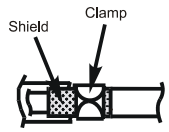
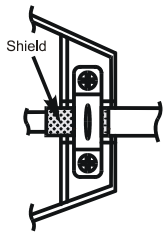
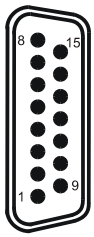
On the HMI side
- 15-pin male SUB-D connector in metallized housing
- Diameter of the cable apt for connecting: 0.25 -0.128 mm^2

Connecting Cable Specifications

- Number of cores: 6
- Core cross-sectional area: 0.14 mm^2
- Cable length: X m
- Max. cable length: 400 m

Cable Shielding

- Complete shielding, no paired shielding
- The shield must be connected to the connector housings on both ends of the cable with the greatest possible surface area (shield twisted, clamped under the strain relief clamps, and wrapped with copper foil).

HMI Cable JN-DK-Xm			
Controller	**Shield**		**HMI**
			
	Connect shield with the greatest possible surface area! Use metallized housing only!		
Pin	**Signal**		**Pin**
6	DC +24 V		15
2	GND		12
5	SDB	RDB	6
7	SDA	RDA	7
3	RDB	SDB	4
1	RDA	SDA	5

Note!

It is recommended to buy the cables from Jetter AG.

15.3.2 HMI Cable KAY_0386-xxxx

Specification of Connector/Socket

On the controller side

- 8-pin miniature DIN male connector
  Type: KMDLA - 8P
  By: KYCON
- Diameter of the cable apt for connecting: 0.128 -0.051 mm^2

On the HMI side
- 15-pin female SUB-D connector in metallized housing
- Diameter of the cable apt for connecting: 0.25 -0.128 mm^2

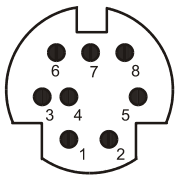
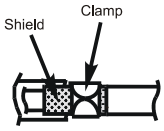
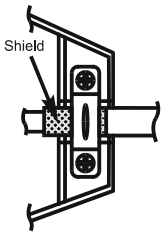
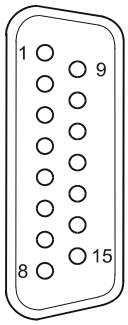
Connecting Cable Specifications

- Number of cores: 5
- Core cross-sectional area: 0.14 mm^2
- Cable length: xxxx cm
- Max. cable length: 400 m

Cable Shielding

- Complete shielding, no paired shielding
- The shield must be connected to the connector housings on both ends of the cable with the greatest possible surface area (shield twisted, clamped under the strain relief clamps, and wrapped with copper foil).

<table>
<tr><th colspan="4">HMI Cable KAY_0386-xxxx</th></tr>
<tr><td>Controller

LCD Interface</td><td colspan="2">Shield</td><td>LCD 60
LCD 62
Port COM 2</td></tr>
<tr><td rowspan="2"> 
</td><td> 
</td><td> 
</td><td rowspan="2"> 
</td></tr>
<tr><td colspan="2">Connect shield with the greatest possible surface area!
Use metallized housing only!</td></tr>
<tr><td>Pin</td><td colspan="2">Signal</td><td>Pin</td></tr>
<tr><td>1</td><td>RDA</td><td>SDA</td><td>14</td></tr>
<tr><td>2</td><td>GND</td><td>GND</td><td>5</td></tr>
<tr><td>3</td><td>RDB</td><td>SDB</td><td>15</td></tr>
<tr><td>5</td><td>SDB</td><td>RDB</td><td>13</td></tr>
<tr><td>7</td><td>SDA</td><td>RDA</td><td>12</td></tr>
<tr><td rowspan="2"></td><td rowspan="2"></td><td rowspan="2">short-circuited [</td><td>11</td></tr>
<tr><td>10</td></tr>
</table>

15.3.3 HMI Cable KAY_0533-0025

The HMIs LCD 52, LCD 54, and LCD 54Z are equipped with a non-detachable cable connecting the LCD with the controller. On the controller side, this cable has a 15-pin male SUB-D connector. For connecting the HMI to the 8-pin Mini-DIN socket of the JC-647 (LCD port) a cable adapter is required.

Specification of Connector/Socket

On the controller side
- 8-pin miniature DIN male connector
  Type: KMDLA - 8P
  By: KYCON
- Diameter of the cable apt for connecting: 0.128 -0.051 mm^2

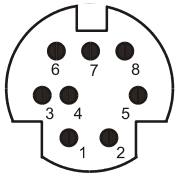
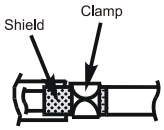
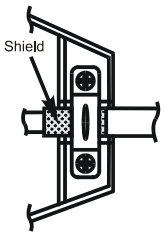
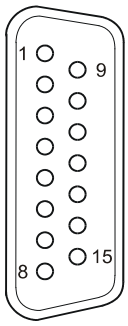
On the HMI side
- 15-pin female SUB-D connector in metallized housing
- Diameter of the cable apt for connecting: 0.25 -0.128 mm^2

Connecting Cable Specifications

- Number of cores: 6
- Core cross-sectional area: 0.14 mm^2
- Cable length: 25 cm

Cable Shielding

- Complete shielding, no paired shielding
- The shield must be connected to the connector housings on both ends of the cable with the greatest possible surface area (shield twisted, clamped under the strain relief clamps, and wrapped with copper foil).

Cable adapter for connecting a LCD 52, LCD 54, and LCD 54Z			
Controller	**Shield**		**LCD Cable**
			
	Connect shield with the greatest possible surface area! Use metallized housing only!		
Pin	**Signal**		**Pin**
6	DC +24 V		4
2	GND		7
5	SDB	RDB	10
7	SDA	RDA	11
3	RDB	SDB	12
1	RDA	SDA	13

Note!

It is recommended to buy the cable adapter from Jetter AG.

15.4 Programming Interface via JETWay-H

Using the JETWay-H interface as programming interface has the following advantages over the RS232 interface:

- Up to 126 PLCs can be addressed from a JetSym workstation. It is advisable to limit the number of nodes to be controlled to 98.
- Transfer rates of up to 115 KBaud can be realized.

15.4.1 Selecting RS-232 or JETWay-H

During system configuration selection is made between programming interface RS232 and JETWay-H in the SYMPAS menu [menu item: Tools / Options / Controller].

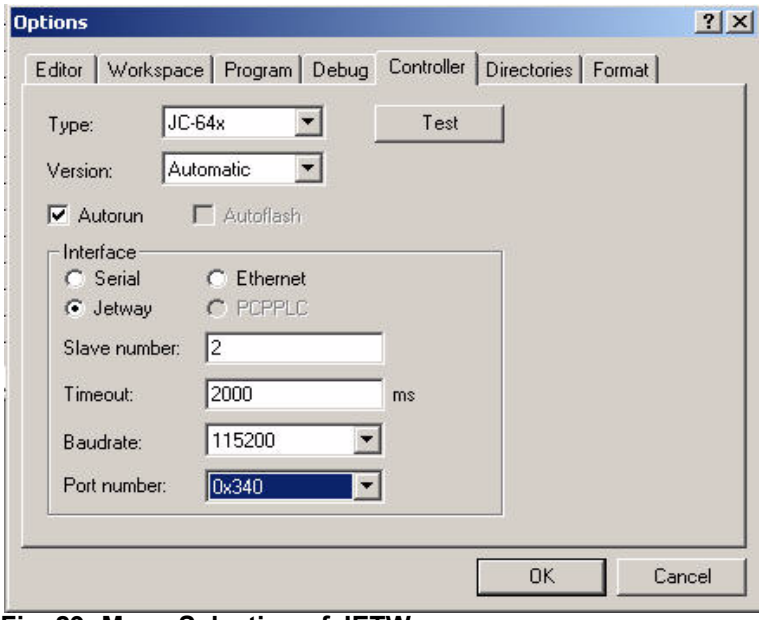


Fig. 29: Menu Selection of JETWay

Additionally, it is possible to enter slave numbers of the controller from 2 through 126.

15.4.2 Description of Connections

Specification of Connectors

On the controller side
- 9-pin male SUB-D connector in metallized housing
- Diameter of the cable apt for connecting: 0.25 -0.128 mm^2

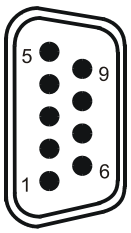
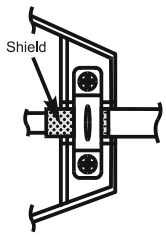
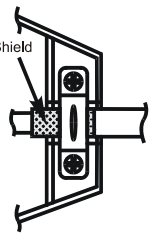
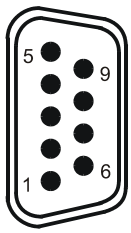
On the PC side
- 9-pin male SUB-D connector in metallized housing
- Diameter of the cable apt for connecting: 0.25 -0.128 mm^2

Connecting Cable Specifications

- Number of cores: 3
- Core cross-sectional area: 0.14 mm^2
- Max. cable length: 400 m

Cable Shielding

- Complete shielding, no paired shielding
- The shield must be connected to the connector housings on both ends of the cable with the greatest possible surface area.

JETWay-H Cable		
Controller	**Shield**	**PC**
	 	
NET1 or NET2	Connect shield with the greatest possible surface area! Use metallized housing only!	**JETWay-H Card NET1 or NET2 of an additional Controller**
Pin	**Signal**	**Pin**
7	GND	7
8	Data +	8
9	Data -	9

15.4.3 JETWay-H PC Card

The connection between JetSym and up to 126 control systems via JETWay-H is realized with the help of the PC card shown below.

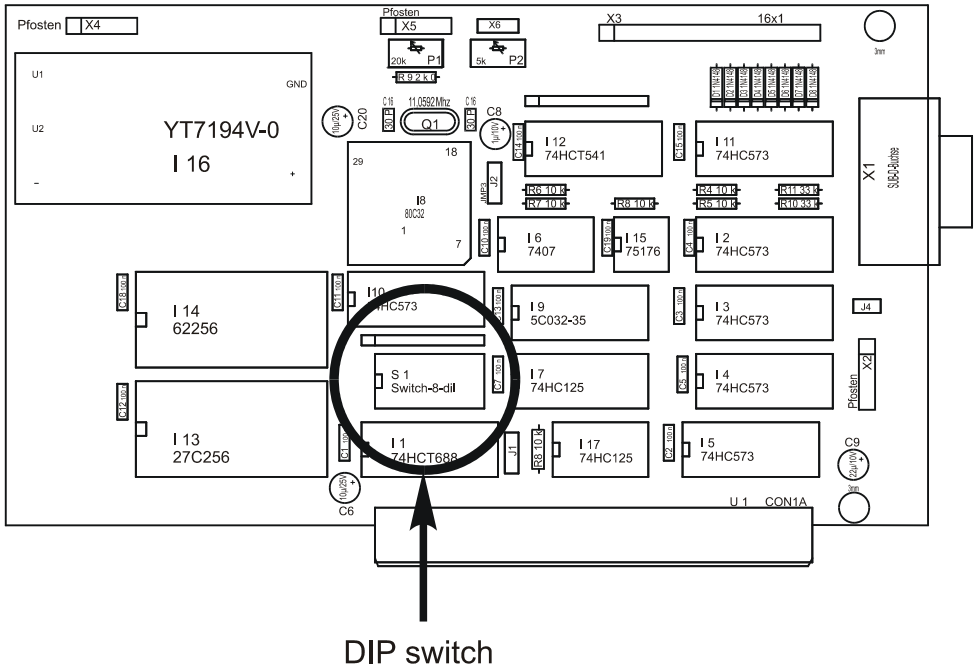


Fig. 30: JETWay-H PC Card

The DIP-switch is used to define the port address. The default setting is 340h.

A different port address can be selected using the DIP-switch on the JETWay-H card as shown above, see Fig. 30.

DIP-switch on the JETWay-H card							
Port x	**S 7**	**S 6**	**S 5**	**S 4**	**S 3**	**S 2**	**S 1**
300h	OFF	OFF	ON	ON	ON	ON	ON
310h	OFF	OFF	ON	ON	ON	OFF	ON
320h	OFF	OFF	ON	ON	OFF	ON	ON
330h	OFF	OFF	ON	ON	OFF	OFF	ON
340h*)	OFF	OFF	ON	OFF	ON	ON	ON
350h	OFF	OFF	ON	OFF	ON	OFF	ON
360h	OFF	OFF	ON	OFF	OFF	ON	ON

*) Default setting

The entry for the port address must be adjusted in JetSym according to the above table.

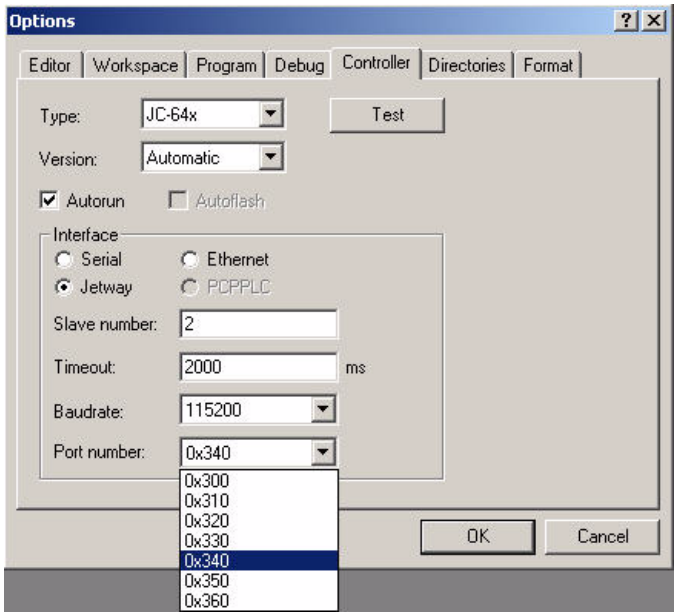


Fig. 31: Setting the Port Address in JetSym

Note!

If the JetSym program is to be used together with the operating system Windows NT and the JETWay port, then the "SETUP JETWAY BOARD" program must be installed.

15.5 Network interface (JETWay-R)

The network interface JETWay-R serves for networking PLCs and/or for the networking of devices, such as remote IOs, valve terminals etc.

Specification of Connectors

On the controller side
- 9-pin male SUB-D connector in metallized housing
- Diameter of the cable apt for connecting: 0.25 -0.128 mm^2

On the other side depending on the connected device

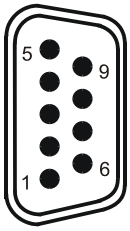
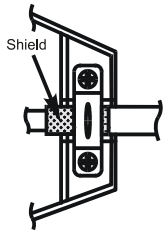
– **Controller or remote D32**
- 9-pin male SUB-D connector in metallized housing
- Diameter of the cable apt for connecting: 0.25 -0.128 mm^2

Connecting Cable Specifications

- Number of cores: 3
- Core cross-sectional area: 0.14 mm^2
- Max. cable length: 400 m

Cable Shielding

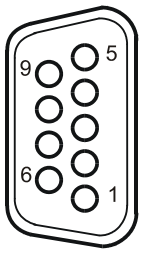
- Complete shielding, no paired shielding
- The shield must be connected to the connector housings on both ends of the cable with the greatest possible surface area.

JETWay-R Cable		
Controller	**Shield**	**Other side of the cable**
		Connector depends on the device to be connected
NET1 or NET2	Connect shield with the greatest possible surface area! Use metallized housing only!	**NET1 or NET2 of an additional controller, remote IO or valve terminal**
Pin	**Signal**	**Pin**
7	GND	7
8	Data +	8
9	Data -	9

15.6 User-Programmable Interface

Network 2 can be configured as user-programmable interface (PRIM).
The following serial interface types are possible:
- RS-232,
- RS-485 (4-wire),
- RS-422,
- RS-485 (2-wire).

NET 2 is the connector socket of network 2.

Pin Assignment of 9-Pin Female Connector SUB-D			
View	**Pin**	**Signal**	**Comment**
	1	-	-
	2	TxD	RS-232
	3	RXD	RS-232
	4	-	-
	5	RDB	RS-422 and RS-485 (4-wire)
	6	RDA	RS-422 / RS-485 (4-wire)
	7	GND	Reference potential
	8	Data + / SDB	RS-485 (2-wire) / RS-422 and RS-485 (4-wire)
	9	Data - / SDA	RS-485 (2-wire) / RS-422 and RS-485 (4-wire)

16 Ethernet Connection

16.1 Connecting cables to the RJ45 Jack

The JC-647 is connected to the Ethernet through its RJ45 jack.

There are two types of interconnecting cables used for 10/100 MBit/s twisted pair Ethernet.

- Straight-through twisted pair cables
- Crossover cables (transmitting and receiving lines are crossed)

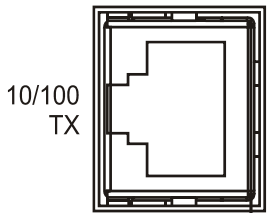


Fig. 32: Ethernet Connection

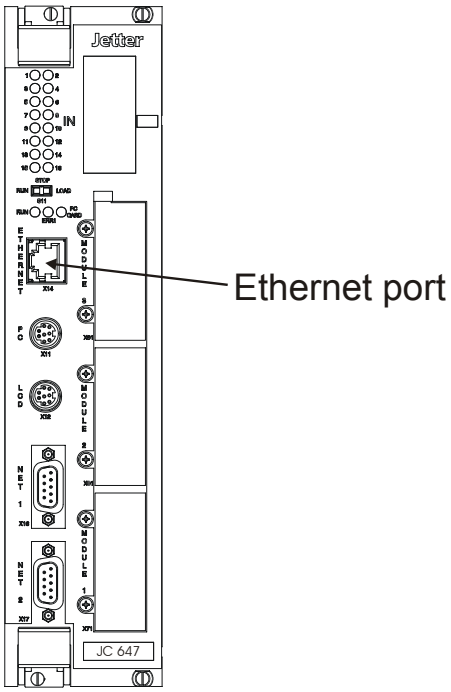


Fig. 33: Ethernet Interface of the JC-647 module

16.2 Connection between JetControl 647 and PC

Direct connection between a PC and a JC-647 is established by means of a crossover cable.

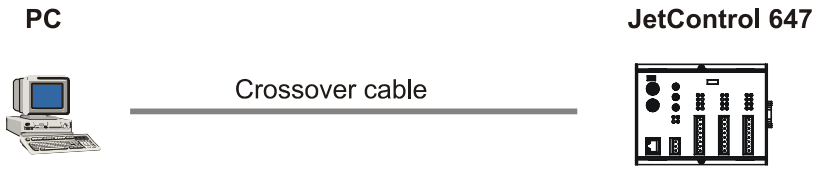


Fig. 34: Ethernet Connection between PC and JetControl

16.3 Connection between JetControl, Switch and PC

If connection between a PC and a JetControl-647 is established through a switch, then straight-through cables have to be used.

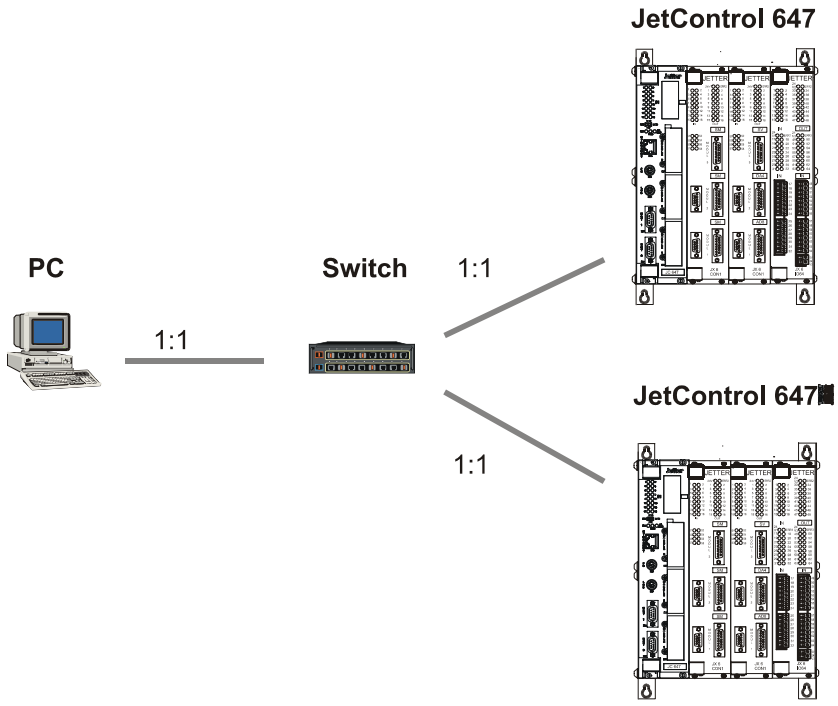


Fig. 35: Ethernet Connection between JetControl and Switch

16.4 Connection via Several Switches

The switches have to be interconnected by means of crossover cables. The connections between JetControl and switch, as well as between PC and switch have to be made through straight-through cables.

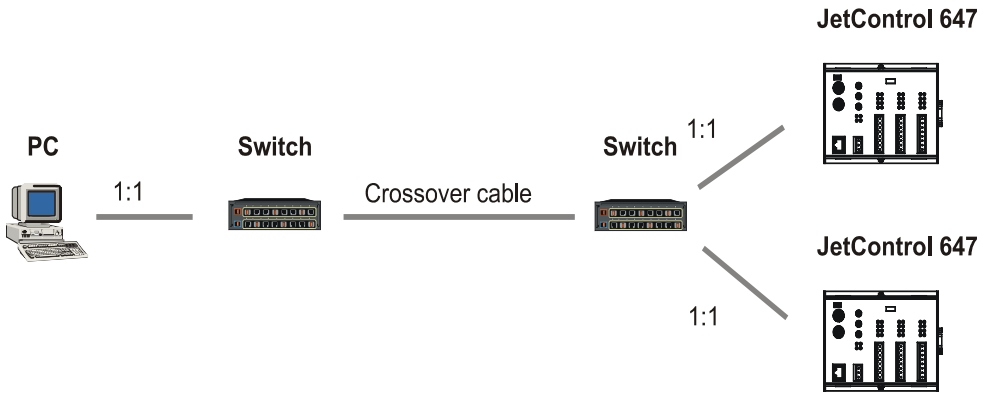


Fig. 36: Ethernet Connection Switch to switch

Note!

Terminals have to be interconnected by means of crossover cables:

- PC to PC
- JetControl to PC
- JetControl to JetControl
- etc.

Terminals with infrastructure components (e.g. switch) have to be interconnected via straight-through cables:

- PC to switch
- JetControl to switch
- etc.

Infrastructure components have to be interconnected by means of crossover cables:

- Switch to switch
- Switch to router
- etc.

Note!

The uplink port of the switch can be used if the proper cable is not available.

17 Software Programming

17.1 Instructions

The following instructions serve to access registers, flags, inputs and outputs.

REGISTER_LOAD, REG	Access to Registers
FLAG	Access to Flags
IN	Access to digital inputs
OUT	Access to digital outputs

Example:

```
          REGISTER_LOAD (100, 1234)          ; Loads 1234 into
                                             ; register 100

          REG 100 = REG 300 + REG 200        ; Adds the content of
                                             ; register 200 to the
                                             ; content of register
                                             ; 300 and stores the
                                             ; result in register
                                             ; 100.

     WHEN FLAG 10                            ; As soon as flag 10
        THEN                                 ; is active, the task
                                             ; will be continued.
          ...

       IF IN 101                             ; If input 101 is
         THEN                                ; active, output 105
           OUT 105                           ; will be set
```

17.2 Access to Inputs or Outputs

Numbering System of Digital Inputs	
Digital Input JC-647	**Address Number**
Input # 1	100 + 1
Input # 2	100 + 2
...	...
Input # 16	100 + 16

Numbering System of Digital Inputs	
Digital Input JX6-I16	**Address Number**
Input # 33	100 + 33
Input # 34	100 + 34
...	...
Input # 47	100 + 47
Input # 48	100 + 48

Numbering system of Digital Outputs	
Digital output JX6-O16	**Address Number**
Output 1	100 + 1
Output 2	100 + 2
...	...
Output 16	100 + 16

17.3 Access to Flags

17.3.1 User Flags

Flags 1 through 2047 are freely available to the user. The flags 256 through 2047 are remanent. The remanent flags are overlaid on registers 0 through 74 such that whole flag ranges can be accessed through registers. There are many possibilities of bit manipulation in conjunction with the instructions for word-processing W-AND, W-OR and W-XOR.

Overlaying of Flags on Registers

Registers	Flags	Comment
0 to 74	1 through 255 256 to 2047 2048 to 2303	Available to the user **Overlay** Special Flags

Example for overlay:

Overlaying of user flags with register 0 as example									
Bit number	0	1	2	3	4	...	21	22	23
Flags	256	257	258	259	260	...	277	278	279

17.3.2 Programming by Means of Flags

Example 1:

A program is to start execution of a process when the start button is pressed and automatic mode is enabled by setting the corresponding flag (e.g. in another task).

```
WHEN IN eStartTaste
     FLAG fAutomaticMode
   THEN
     ...
```

Example 2:

Execution of a second task, the automatic task, is to be started in the main task using a flag.

```
TASK tMainTask----------------------------
      ...
        IF
            IN iStartButton
          THEN
            FLAG fAutomaticMode
      ...
            GOTO tMainTask
TASK tAutomaticMode-----------------------
-----
      WHEN
            FLAG fAutomaticMode
           THEN
            ...
            GOTO tAutomaticMode
```

17.4 Access to Registers

216987 remanent user registers are available to the user in the register areas 100 through 20479 and 196608 through 393215. They serve as buffers for storage of comparison and measured values, as well as of setpoints.

These registers are 32 bits wide and have a value range from -2,147,483,648 through +2,147,483,647.

Note!

The contents of the 216987 user registers are maintained after switching off the power supply.

17.4.1 Programming by Means of Registers

The instruction `REGISTER_LOAD [ x, a ]` serves for loading numeric values (or contents of other registers) into a register.

Description:

In the instruction above, "x" represents the number of the register into which value "a" is to be written by analogy with Fig. 37 and Fig. 38.

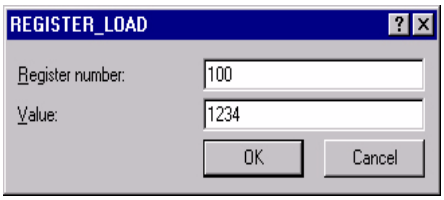


Fig. 37: REGISTER_LOAD with numeric parameters

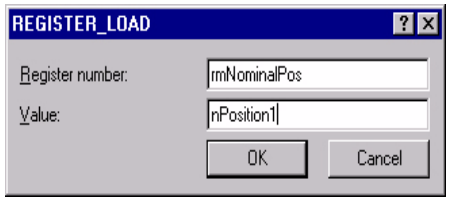


Fig. 38: REGISTER_LOAD with symbolic parameters

JetSym ST

Using the assingment operator ":=" the numerical value is stored to the specified register.

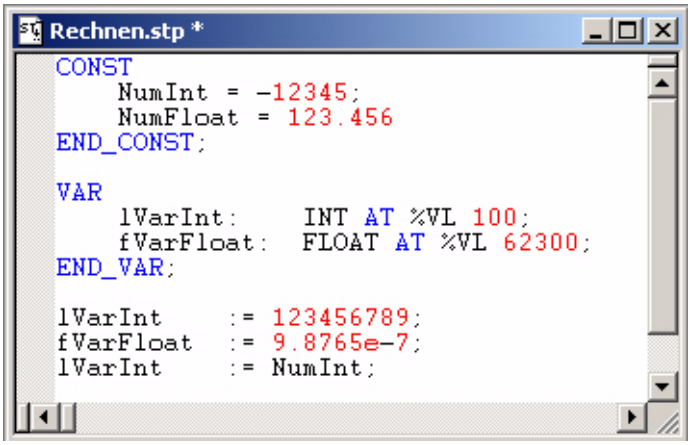


Fig. 39: Assigning a constant to a variable (JetSym ST)

The numerical value "123456789" is loaded into the variable located at 100 ("Register 100"). Then, the numerical value "9.8765 10-7" is loaded into the floating point variable located at 65024 ("Register 65024"). Then, value "-12345" is assigned to "register 100".

Indirect and Double Indirect Addressing

JetSym

For the "x" and the "a" in the instruction shown above, not only a number can be written but a register can be specified as well. To do so, prefix an “@“ to the register number by pressing the space key.
If “@(y)“ is written instead of "x", then value "a" is written into the register the number of which is contained in register "y".
If “@(b)“ is written instead of "a", then not the value itself but the content of the specified register is loaded into register "x" (or @(y)).
If “@@(b)“ is written instead of "a", the result is as follows: First, the value of the register with number b is read. This value then serves as register number. This means, a new value is read in the register with the specified number, and the new value is then stored to register x.

Fig. 40: Indirect and Double Indirect Addressing (JetSym)

JetSym ST

With JetSym ST, for this purpose the address operator "&" and the pointer operator "@" are used.

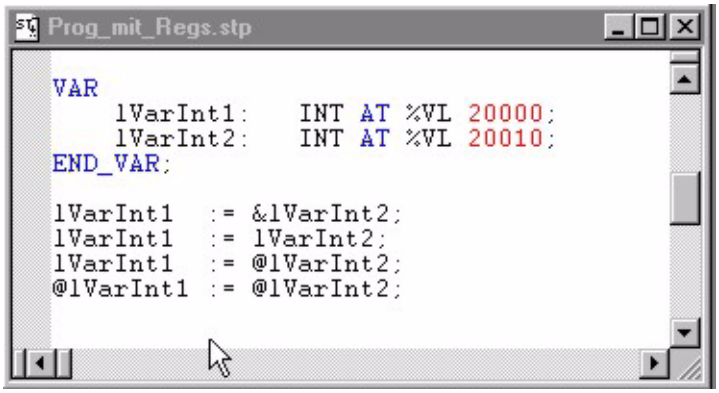


Fig. 41: Indirect and Double Indirect Addressing (JetSym ST)

First, the address of the variable "1VarInt2" (here: 20010) is assigned to the variable "1VarInt1". Then, the content of "1VarInt2" is copied to "1VarInt1". Then, the content of the "register", to which "1VarInt2" is pointing (the number of which is contained in "1VarInt2"), is copied to "1VarInt1". Finally, the content of the "register", to which "1VarInt2" is pointing is copied to register, to which "1VarInt1" is pointing.

Examples:

1. Loading a number into a register

JetSym	`REGISTER_LOAD (rNewPosition, 1280)`
JetSym ST	`rNewPosition := 1280;`

Value 1280 is loaded into the register **`rNewPosition`**.

2. Copying one register into another register

JetSym	`REGISTER_LOAD (rVoltage, @rVoltage1)`
JetSym ST	`rVoltage := rVoltage1;`

The value in register **`rVoltage1`** will be copied into register **`rVoltage`**. In other words, the contents of register rVoltage1 is copied into register rVoltage.

3. Loading by double indirect addressing

JetSym
```
REGISTER_LOAD (rVoltage, @@rVPointer)
```

JetSym ST
```
rVoltage := @@rVPointer;
```

The value of the register, the number of which is specified in the register **rVPointer**, is loaded into register **rVoltage**.

4. Double Indirect Addressing: Numerical example

Register Number	Value
REG 64	111
REG 111	70035
REG 150	11
REG 11	Any value

This example will be carried out using the following instruction:

JetSym
```
REGISTER_LOAD (@150, @@64)
```

JetSym ST
```
VAR
pDestination: INT AT %VL 150;
pSource: INT AT %VL 64;
END_VAR;

@pDestination := @pSource;
```

This instruction results in the following register values and the graphic representation shown in Fig. 42:

Register 64 = 111 remains unchanged

Register 111 = 70035 remains unchanged

Register 150 = 11 remains unchanged

Register 11 = @150 = @@64 = @111 = 70035

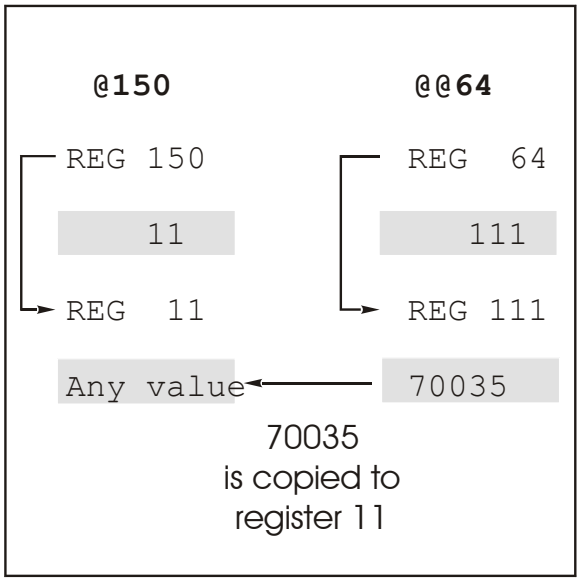


Fig. 42: Example for Double Indirect Addressing with JetSym

17.4.2 Calculations by Means of Registers

For forming expressions, in JetSym und JetSym ST the arithmetical operators for the 4 basic arithmetical operations (+, -, *, /), and the operators for bit-wise interconnection of registers (WAND, WOR, WXOR) are available.

Processing expressions

JetSym

Expressions are strictly processed from left to right (no "order of operation rule"). If a specific order of operations is to be ensured, brackets have to be inserted manually.

JetSym ST

Here, multiplication, division and bit-wise AND relation are given priority over addition, subtraction and bit-wise OR, and XOR relations. If there is no priority, the expression is processed from left to right.

Note!

An operation is given priority by means of brackets which are automatically inserted. This fact has to be taken into account when processing expressions where floating point and integer values are mixed (for more information refer to the next section).

Calculations of Floating Point / Integer Values

Basically, floating point and integer values and constants can be mixed within an arithmetical expression. However, especially with value overruns or underruns the prinicple of operation of the OS has to be taken into account:

- Processing of an expression is always started as integer calculation. This means that intermediate results of divisions are stored without decimal places.
- When the first floating point register or floating point constant is encounterd, processing switches over to floating point operation. The remaining expression is calculated using floating point values (integer values will also be converted).
- With an opening bracket and a function call, the existing intermediate result, as well as the calculation type are internally buffered, and the operating system switches over to integer calculation.
- With a closing bracket and when returning from a function, the OS resumes the buffered calculation type.
- When comparing arithmetical expressions, the calculation of the expression "on the right side of the equals sign" is started as integer calculation, irrespective of how the calculation of the expression "on the left side of the equals sign" has been carried out.
- With an assignment, the calculation of the expression "on the right side of the assignment operator" starts with the type of register (variable) to which the result is to be assigned, provided the register (variable) is a local JetControl 24x register (variable). Otherwise, an integer calculation starts.
- When a floating point value is assigned to an integer register (integer variable), the value is rounded to the next integer.
- Operators for bit-wise relations may be used for integer calculations only.

Examples:

JetSym
```
REG rResult = 12 / 5 * 2
```

JetSym ST
```
rResult := 12 / 5 * 2;
```

- If "rResult" is an integer register or a register located on an expansion module, value 4 is assigned to it, since decimal places are not taken into account with integer divisions.
- If "rResult" is a local floating point register, calculations are carried out using floating point values. Thus, the result, which is assigned to the register, is 4.8.

JetSym
```
REG rResult = 12.0 / 5 * 2
```

JetSym ST
```
rResult := 12.0 / 5 * 2;
```

- If "rResult" is an integer register, the value 5 is assigned to it, since the floating point constant causes a switchover to floating point calculations. Then, the result of the calculation (4.8) is rounded up as it is assigned to an integer register.
- If "rResult" is a floating point register (local or located on an expansion module), the value 4.8 is assigned.

JetSym

In JetSym the following instructions are used for calculations:

– **REG <RegNo>**
– **REGZERO <RegNo>**
– **REGDEC <RegNo>**
– **REGINC <RegNo>**
– **SHIFT_LEFT (<RegNo> <Amount>)**
– **SHIFT_RIGHT (<RegNo> <Amount>)**

Note!

I is possible for all six instructions to specify the register number indirectly (which is the only parameter to be specified).

The instructions SHIFT_LEFT and SHIFT_RIGHT are new ones.

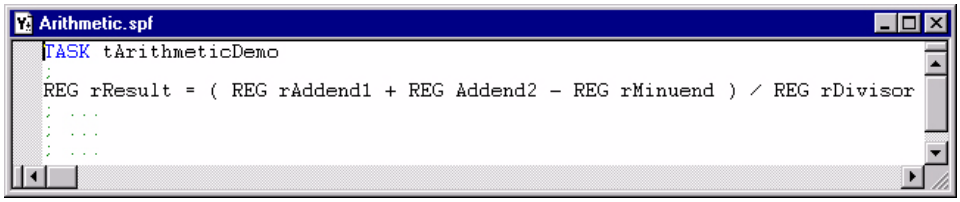


Fig. 43: Example of Register Arithmetic (JetSym)

The REG <RegNo> Instruction

This instruction obtains direct access to the value of a register and can be dealt with like a variable. In an output instruction, a certain value is assigned to the register left to the equals sign.
In an input condition, the content of the register is read and no value is assigned. In both cases, the register accesses right to the equals sign result in reading the register content (output instruction or input condition).

Example 1:

```
THEN
  REG 1 = REG 105 * 25
```

This example shows an assignment (output instruction, which is initiated by THEN). By doing so, register 105 is read and its content is multiplied by 25. The result of this arithmetic operation is stored in register 1. The content of register 105 remains unchanged.

Example 2:

```
IF REG 1 = REG 105 * 25
  THEN
```

In this case, the expression `REG 1 = REG 105 * 25` is not part of an output instruction but of an input condition. In this part of the program, the value of register 1 remains unchanged. It will only be compared with the product `REG 105 * 25`.

The REGZERO <RegNo> Instruction

The REGZERO instruction sets a register to zero (as output instruction) or queries whether the content of a register is zero (as input condition).

After IF or WHEN, this instruction has the following meaning as input condition. An explanation will be given in the following example.

Example:

```
IF REGZERO 49                    IF REG 49 = 0
  THEN                             THEN
```

These two program parts have got the same function.
On the right-hand side of the example the comparison is carried out as general arithmetic comparison. On the left-hand side of the example the special instruction **REGZERO** is used.
Using **REGZERO** speeds up program execution.

The REGINC <RegNo> Instruction

This instruction serves for increasing the content of a register (increment).
This function is frequently used in loops to increment counters and pointers.

Example:

```
THEN                             THEN
  REGINC 88                        REG 88 = REG 88 + 1
```

The REGDEC <RegNo> Instruction

This instruction serves for decreasing the content of a register (decrement).
This function is frequently used in loops to decrement counters and pointers.

Example 1:

```
  THEN                                  THEN
    REGDEC 100                            REG 100 = REG 100 -1
```

Example 2:

```
      REGISTER_LOAD (1, 10)
LABEL 55
      ...
      REGDEC 1
  IF REGZERO 1
    THEN
    ELSE
      GOTO 55
    THEN
```

This way, a loop can be realized which executes a certain number of iterations. During each run of the loop, the value of the "counting register" is decremented by one and is being checked whether it is 0 (REGZERO 1). If the value is zero, the first THEN will be ignored and the loop will go to the second THEN to continue execution of the program there. If the value of register 1 is not zero, the program will go back to the starting point of the loop.

SHIFT_LEFT (<RegNo> <Amount>) Instruction

By means of this instruction, register contents can bitwise be shifted to the left (which corresponds to a multiplication by 2).
To do so, you have to specify the amount of bits to be shifted (from 1 to 32).
The bit shifted out of the register last can be read from special flag 2076. Zero bits are shifted into the register from the other side.

An amount of 0 bits will not affect the register and flag 2076.

The register number and the number of shift operations can be specified as parameters directly and indirectly.

In example 1 a demonstration of direct addressing is given:

Example 1:

```
THEN
  SHIFT_LEFT (100, 3)     ; The contents of register 100 is
                            three times shifted bitwise to the
                            left.
```

This example shows a way how to multiply the contents of register 100 by 8. This approach has the advantage over an arithmetic expression of speeding up processing.

In example 2 a demonstration of indirect addressing is given:

Example 2:

```
THEN
  SHIFT_LEFT (@200, @201) ; The contents of register 200
                            indicate the register, the
                            contents of which is to be shifted
                            to the left.
                            The contents of register 201
                            indicates the number of shift
                            operations.
```

SHIFT_RIGHT (<RegNo> <Amount>) Instruction

By means of this instruction, register contents can bitwise be shifted to the right (which corresponds to a division by 2).
To do so, you have to specify the amount of bits to be shifted (from 1 to 32).
The bit shifted out of the register last can be read from special flag 2076. Zero bits are shifted into the register from the other side.

An amount of 0 bits will not affect the register and flag 2076.

Example 1:

```
THEN
  SHIFT_RIGHT (100, 1)        ; The contents of register 100 is one
                                time shifted bitwise to the right.
  IF FLAG 2076 THEN
```

If these instructions are used in a loop, it is very easy to make execution of a processing step conditional on the state of individual bits of register 100.

JetSym ST

With JetSym ST the following instructions are available:

- **DEC (<RegNo>)**
- **INC (<RegNo>)**
- **SHIFT_LEFT (<RegNo>, <BitNo>)**
- **SHIFT_RIGHT (<RegNo> <BitNo>)**

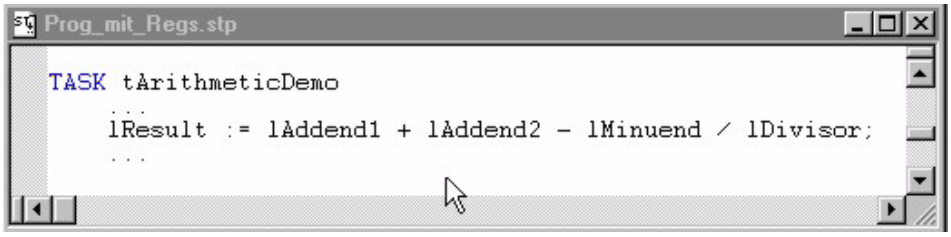


Fig. 44: Example of Register Arithmetic (JetSym ST)

The INC (<RegNo>) Instruction

This instruction serves for increasing the content of a register (increment).
This function is frequently used in loops to increment counters and pointers.

Example:

```
THEN                          THEN
  INC (lCounter);               lCounter := lCounter + 1;
```

These two program parts have got the same function. With both of them, the content of variable "1Counter" is incremented by 1.

The DEC (<RegNo>) Instruction

This instruction serves for decreasing the content of a register (decrement). This function is frequently used in loops to decrement counters and pointers.

Example 1:

```
THEN
  DEC (lCounter);
```

```
THEN
  lCounter := lCounter -1;
```

These two program parts have got the same function. With both of them, the content of variable "1Counter" is decremented by 1.

Example 2:

```
lCounter := 10;
WHILE lCounter DO
  ...
  DEC (lCounter);
  ...
END_WHILE;
```

This way, a loop can be realised which executes a certain number of iterations. At the beginning of the loop, a check is made whether the "counting register" is unequal to "0". If this is the case, the instructions within the loop are executed, the counter is decremented by one and a jump to the beginning of the loop is made. Once the counter is "0", the program starts to execute the instruction behind "END_WHILE".

Note!

A loop with a certain number of cycles could also be programmed using the instruction FOR.

The instructions SHIFT_LEFT and SHIFT_RIGHT

These to instructions are used to shift all bits of a register by a user-definable number of places to the right or left. Shifting a bit by one position corresponds to a multiplication (_LEFT) or a division (_RIGHT) by 2.
These instructions are, for example, used to assign information to a part when transporting it through a plant.
The bit shifted out of the register last can be read out of flag 2076.

Example 1:

```
THEN
  SHIFT_LEFT (lVarl, 3);
```

This example shows a way how to multiply the contents of "1Var1" by 8. This approach has the advantage over an arithmetic expression of speeding up processing.

Example 2:

```
THEN
  SHIFT_RIGHT (lDrill, 1);
  IF bfCarry THEN                // flag 2076
  ...
```

If these instructions are used in a loop, it is very easy to make execution of a processing step conditional on the state of individual bits of the variable "1Drill".

17.4.3 Accessing Digital Inputs Using Registers

A group of combined digital inputs can be addressed by accessing the special registers listed in the following table.

Register 62592 serves as example:

- The instruction REGISTER_LOAD (100, @62592) is used to read the state of the digital inputs 101 through 124.

24 Combined Digital Inputs by the example of register 62592									
Bit number	0	1	2	3	4	...	21	22	23
Input	101	102	103	104	105	...	122	123	124

24 Combined Inputs	
Registers	
62592	Inputs 101 .. 124
62593	Inputs 109 .. 132
62594	Inputs 117 .. 140
62595	Inputs 125 .. 148
62596	Inputs 133 .. 156
62597	Inputs 141 .. 164
62598	Inputs 149 .. 164
62599	Inputs 157 .. 164
62600	Inputs 201 .. 224
62601	Inputs 209 .. 232
62602	Inputs 217 .. 240
62603	Inputs 225 .. 248
62604	Inputs 233 .. 256
62605	Inputs 241 .. 264
62606	Inputs 249 .. 264
62607	Inputs 257 .. 264
62608	Inputs 301 .. 324
62609	Inputs 309 .. 332
62610	Inputs 317 .. 340
62611	Inputs 325 .. 348
62612	Inputs 333 .. 356
62613	Inputs 341 .. 364
62614	Inputs 349 .. 364
62615	Inputs 357 .. 364
62616	Inputs 401 .. 424
62617	Inputs 409 .. 432
62618	Inputs 417 .. 440
62619	Inputs 425 .. 448
62620	Inputs 433 .. 456

62621	Inputs 441 .. 464
62622	Inputs 449 .. 464
62623	Inputs 457 .. 464
62624	Inputs 501 .. 524
62625	Inputs 509 .. 532
62626	Inputs 517 .. 540
62627	Inputs 525 .. 548
62628	Inputs 533 .. 556
62629	Inputs 541 .. 564
62630	Inputs 549 .. 564
62631	Inputs 557 .. 564
62632	Inputs 601 .. 624
62633	Inputs 609 .. 632
62634	Inputs 617 .. 640
62635	Inputs 625 .. 648
62636	Inputs 633 .. 656
62637	Inputs 641 .. 664
62638	Inputs 649 .. 664
62639	Inputs 657 .. 664
62640	Inputs 701 .. 724
62641	Inputs 709 .. 732
62642	Inputs 717 .. 740
62643	Inputs 725 .. 748
62644	Inputs 733 .. 756
62645	Inputs 741 .. 764
62646	Inputs 749 .. 764
62647	Inputs 757 .. 764
62648	Inputs 801 .. 824
62649	Inputs 809 .. 832
62650	Inputs 817 .. 840
62651	Inputs 825 .. 848
62652	Inputs 833 .. 856
62653	Inputs 841 .. 864
62654	Inputs 849 .. 864
62655	Inputs 857 .. 864

16 Combined Digital Inputs by the example of register 62528									
Bit number	0	1	2	3	4	...	13	14	15
Input	101	102	103	104	105	...	114	115	116

16 Combined Inputs	
Registers	
62528	Inputs 101 .. 116
62529	Inputs 109 .. 124
62530	Inputs 117 .. 132
62531	Inputs 125 .. 140
62532	Inputs 133 .. 148
62533	Inputs 141 .. 156
62534	Inputs 149 .. 164
62535	Inputs 157 .. 164
62536	Inputs 201 .. 216
62537	Inputs 209 .. 224
62538	Inputs 217 .. 232
62539	Inputs 225 .. 240
62540	Inputs 233 .. 248
62541	Inputs 241 .. 256
62542	Inputs 249 .. 264
62543	Inputs 257 .. 264
62544	Inputs 301 .. 316
62545	Inputs 309 .. 324
62546	Inputs 317 .. 332
62547	Inputs 325 .. 340
62548	Inputs 333 .. 348
62549	Inputs 341 .. 356
62550	Inputs 349 .. 364
62551	Inputs 357 .. 364
62552	Inputs 401 .. 416
62553	Inputs 409 .. 424
62554	Inputs 417 .. 432
62555	Inputs 425 .. 440
62556	Inputs 433 .. 448
62557	Inputs 441 .. 456
62558	Inputs 449 .. 464
62559	Inputs 457 .. 464
62560	Inputs 501 .. 516
62561	Inputs 509 .. 524
62562	Inputs 517 .. 532
62563	Inputs 525 .. 540
62564	Inputs 533 .. 548

62565	Inputs 541 .. 556
62566	Inputs 549 .. 564
62567	Inputs 557 .. 564
62568	Inputs 601 .. 616
62569	Inputs 609 .. 624
62570	Inputs 617 .. 632
62571	Inputs 625 .. 640
62572	Inputs 633 .. 648
62573	Inputs 641 .. 656
62574	Inputs 649 .. 664
62575	Inputs 657 .. 664
62576	Inputs 701 .. 716
62577	Inputs 709 .. 724
62578	Inputs 717 .. 732
62579	Inputs 725 .. 740
62580	Inputs 733 .. 748
62581	Inputs 741 .. 756
62582	Inputs 749 .. 764
62583	Inputs 757 .. 764
62584	Inputs 801 .. 816
62585	Inputs 809 .. 824
62586	Inputs 817 .. 832
62587	Inputs 825 .. 840
62588	Inputs 833 .. 848
62589	Inputs 841 .. 856
62590	Inputs 849 .. 864
62591	Inputs 857 .. 864

8 Combined Digital Inputs by the example of register 62464								
Bit number	0	1	2	3	4	5	6	7
Input	101	102	103	104	105	106	107	108

8 Combined Inputs	
Registers	
62464	Inputs 101 .. 108
62465	Inputs 109 .. 116
62466	Inputs 117 .. 124
62467	Inputs 125 .. 132
62468	Inputs 133 .. 140
62469	Inputs 141 .. 148
62470	Inputs 149 .. 156
62471	Inputs 157 .. 164
62472	Inputs 201 .. 208
62473	Inputs 209 .. 216
62474	Inputs 217 .. 224
62475	Inputs 225 .. 232
62476	Inputs 233 .. 240
62477	Inputs 241 .. 248
62478	Inputs 249 .. 256
62479	Inputs 257 .. 264
62480	Inputs 301 .. 308
62481	Inputs 309 .. 316
62482	Inputs 317 .. 324
62483	Inputs 325 .. 332
62484	Inputs 333 .. 340
62485	Inputs 341 .. 348
62486	Inputs 349 .. 356
62487	Inputs 357 .. 364
62488	Inputs 401 .. 408
62489	Inputs 409 .. 416
62490	Inputs 417 .. 424
62491	Inputs 425 .. 432
62492	Inputs 433 .. 440
62493	Inputs 441 .. 448
62494	Inputs 449 .. 456
62495	Inputs 457 .. 464
62496	Inputs 501 .. 508
62497	Inputs 509 .. 516
62498	Inputs 517 .. 524
62499	Inputs 525 .. 532
62500	Inputs 533 .. 540

62501	Inputs 541 .. 548
62502	Inputs 549 .. 556
62503	Inputs 557 .. 564
62504	Inputs 601 .. 608
62505	Inputs 609 .. 616
62506	Inputs 617 .. 624
62507	Inputs 625 .. 632
62508	Inputs 633 .. 640
62509	Inputs 641 .. 648
62510	Inputs 649 .. 656
62511	Inputs 657 .. 664
62512	Inputs 701 .. 708
62513	Inputs 709 .. 716
62514	Inputs 717 .. 724
62515	Inputs 725 .. 732
62516	Inputs 733 .. 740
62517	Inputs 741 .. 748
62518	Inputs 749 .. 756
62519	Inputs 757 .. 764
62520	Inputs 801 .. 808
62521	Inputs 809 .. 816
62522	Inputs 817 .. 824
62523	Inputs 825 .. 832
62524	Inputs 833 .. 840
62525	Inputs 841 .. 848
62526	Inputs 849 .. 856
62527	Inputs 857 .. 864

17.4.4 Accessing Digital Outputs Using Registers

A group of combined digital outputs can be addressed by accessing the special registers listed in the following table.

Register 62848 serves as example:

- The instruction REGISTER_LOAD (62848, a) is used to make the changes specified by the value a to the state of the digital inputs 101 through 116 .

24 Combined Digital Outputs by the example of register 62848									
Bit number	0	1	2	3	4	...	21	22	23
Output	101	102	103	104	105	...	122	123	124

24 Combined Outputs	
Registers	
62848	Outputs 101 .. 124
62849	Outputs 109 .. 132
62850	Outputs 117 .. 140
62851	Outputs 125 .. 148
62852	Outputs 133 .. 156
62853	Outputs 141 .. 164
62854	Outputs 149 .. 164
62855	Outputs 157 .. 164
62856	Outputs 201 .. 224
62857	Outputs 209 .. 232
62858	Outputs 217 .. 240
62859	Outputs 225 .. 248
62860	Outputs 233 .. 256
62861	Outputs 241 .. 264
62862	Outputs 249 .. 264
62863	Outputs 257 .. 264
62864	Outputs 301 .. 324
62865	Outputs 309 .. 332
62866	Outputs 317 .. 340
62867	Outputs 325 .. 348
62868	Outputs 333 .. 356
62869	Outputs 341 .. 364
62870	Outputs 349 .. 364
62871	Outputs 357 .. 364
62872	Outputs 401 .. 424
62873	Outputs 409 .. 432
62874	Outputs 417 .. 440

62875	Outputs 425 .. 448
62876	Outputs 433 .. 456
62877	Outputs 441 .. 464
62878	Outputs 449 .. 464
62879	Outputs 457 .. 464
62880	Outputs 501 .. 524
62881	Outputs 509 .. 532
62882	Outputs 517 .. 540
62883	Outputs 525 .. 548
62884	Outputs 533 .. 556
62885	Outputs 541 .. 564
62886	Outputs 549 .. 564
62887	Outputs 557 .. 564
62888	Outputs 601 .. 624
62889	Outputs 609 .. 632
62890	Outputs 617 .. 640
62891	Outputs 625 .. 648
62892	Outputs 633 .. 656
62893	Outputs 641 .. 664
62894	Outputs 649 .. 664
62895	Outputs 657 .. 664
62896	Outputs 701 .. 724
62897	Outputs 709 .. 732
62898	Outputs 717 .. 740
62899	Outputs 725 .. 748
62900	Outputs 733 .. 756
62901	Outputs 741 .. 764
62902	Outputs 749 .. 764
62903	Outputs 757 .. 764
62904	Outputs 801 .. 824
62905	Outputs 809 .. 832
62906	Outputs 817 .. 840
62907	Outputs 825 .. 848
62908	Outputs 833 .. 856
62909	Outputs 841 .. 864
62910	Outputs 849 .. 864
62911	Outputs 857 .. 864

16 Combined Digital Outputs by the example of register 62784									
Bit number	0	1	2	3	4	...	13	14	15
Output	101	102	103	104	105	...	114	115	116

16 Combined Outputs	
Registers	
62784	Outputs 101 .. 116
62785	Outputs 109 .. 124
62786	Outputs 117 .. 132
62787	Outputs 125 .. 140
62788	Outputs 133 .. 148
62789	Outputs 141 .. 156
62790	Outputs 149 .. 164
62791	Outputs 157 .. 164
62792	Outputs 201 .. 216
62793	Outputs 209 .. 224
62794	Outputs 217 .. 232
62795	Outputs 225 .. 240
62796	Outputs 233 .. 248
62797	Outputs 241 .. 256
62798	Outputs 249 .. 264
62799	Outputs 257 .. 264
62800	Outputs 301 .. 316
62801	Outputs 309 .. 324
62802	Outputs 317 .. 332
62803	Outputs 325 .. 340
62804	Outputs 333 .. 348
62805	Outputs 341 .. 356
62806	Outputs 349 .. 364
62807	Outputs 357 .. 364
62808	Outputs 401 .. 416
62809	Outputs 409 .. 424
62810	Outputs 417 .. 432
62811	Outputs 425 .. 440
62812	Outputs 433 .. 448
62813	Outputs 441 .. 456
62814	Outputs 449 .. 464
62815	Outputs 457 .. 464
62816	Outputs 501 .. 516
62817	Outputs 509 .. 524
62818	Outputs 517 .. 532
62819	Outputs 525 .. 540
62820	Outputs 533 .. 548

62821	Outputs 541 .. 556
62822	Outputs 549 .. 564
62823	Outputs 557 .. 564
62824	Outputs 601 .. 616
62825	Outputs 609 .. 624
62826	Outputs 617 .. 632
62827	Outputs 625 .. 640
62828	Outputs 633 .. 648
62829	Outputs 641 .. 656
62830	Outputs 649 .. 664
62831	Outputs 657 .. 664
62832	Outputs 701 .. 716
62833	Outputs 709 .. 724
62834	Outputs 717 .. 732
62835	Outputs 725 .. 740
62836	Outputs 733 .. 748
62837	Outputs 741 .. 756
62838	Outputs 749 .. 764
62839	Outputs 757 .. 764
62840	Outputs 801 .. 816
62841	Outputs 809 .. 824
62842	Outputs 817 .. 832
62843	Outputs 825 .. 840
62844	Outputs 833 .. 848
62845	Outputs 841 .. 856
62846	Outputs 849 .. 864
62847	Outputs 857 .. 864

8 Combined Digital Outputs by the example of register 62720								
Bit number	0	1	2	3	4	5	6	7
Output	101	102	103	104	105	106	107	108

8 Combined Outputs	
Registers	
62720	Outputs 101 .. 108
62721	Outputs 109 .. 116
62722	Outputs 117 .. 124
62723	Outputs 125 .. 132
62724	Outputs 133 .. 140
62725	Outputs 141 .. 148
62726	Outputs 149 .. 156
62727	Outputs 157 .. 164
62728	Outputs 201 .. 208
62729	Outputs 209 .. 216
62730	Outputs 217 .. 224
62731	Outputs 225 .. 232
62732	Outputs 233 .. 240
62733	Outputs 241 .. 248
62734	Outputs 249 .. 256
62735	Outputs 257 .. 264
62736	Outputs 301 .. 308
62737	Outputs 309 .. 316
62738	Outputs 317 .. 324
62739	Outputs 325 .. 332
62740	Outputs 333 .. 340
62741	Outputs 341 .. 348
62742	Outputs 349 .. 356
62743	Outputs 357 .. 364
62744	Outputs 401 .. 408
62745	Outputs 409 .. 416
62746	Outputs 417 .. 424
62747	Outputs 425 .. 432
62748	Outputs 433 .. 440
62749	Outputs 441 .. 448
62750	Outputs 449 .. 456
62751	Outputs 457 .. 464
62752	Outputs 501 .. 508
62753	Outputs 509 .. 516
62754	Outputs 517 .. 524
62755	Outputs 525 .. 532
62756	Outputs 533 .. 540

62757	Outputs 541 .. 548
62758	Outputs 549 .. 556
62759	Outputs 557 .. 564
62760	Outputs 601 .. 608
62761	Outputs 609 .. 616
62762	Outputs 617 .. 624
62763	Outputs 625 .. 632
62764	Outputs 633 .. 640
62765	Outputs 641 .. 648
62766	Outputs 649 .. 656
62767	Outputs 657 .. 664
62768	Outputs 701 .. 708
62769	Outputs 709 .. 716
62770	Outputs 717 .. 724
62771	Outputs 725 .. 732
62772	Outputs 733 .. 740
62773	Outputs 741 .. 748
62774	Outputs 749 .. 756
62775	Outputs 757 .. 764
62776	Outputs 801 .. 808
62777	Outputs 809 .. 816
62778	Outputs 817 .. 824
62779	Outputs 825 .. 832
62780	Outputs 833 .. 840
62781	Outputs 841 .. 848
62782	Outputs 849 .. 856
62783	Outputs 857 .. 864

17.5 Positioning Instructions

The following instructions are used for positioning tasks:

JetSym

- **POS <AxisAddress> <Destination> <Speed>**
- **POS <AxisAddress>**
- **ACTUAL_POS <AxisAddress>**

JetSym ST

- **POS (>AxisAddress<, <Destination>, <Speed>);**
- **AXARR (<AxisAddress>);**
- **ACTUAL_POS (<AxisAddress>);**

Note!

Supplement:

The above macro instructions can be used to address JX2-Slave modules via a JX6-SB-I submodule.

The axis address is coded as follows:

Coding of the axis address: mxy

- The first digit **m** specifies the JetControl 647 **submodule socket** where the JX6-SB-I submodule is located..
  m = Submodule socket (1 .. 3).

- The second digit **x** specifies the **number of the slave module** connected to the system bus:
  x = Slave module number (2 .. 9).

- The third digit **y** is the **number of the axis or controller** (0 .. 9).

18 HMIs, Operator Guidance

18.1 Technical Data

Overview of HMIs (HMIs)				
Type	**Display**	**Keys**	**Comment**	**Interface cable**
LCD 9	2 lines of 24 characters each	– 12 F keys (with LED) – Special Function Keys – Numeric keypad		OpenColl JN-DK Page 90
LCD 10	2 lines of 24 characters each	– 12 F keys (with LED) – Special Function Keys – Numeric keypad	9 mm Character height backlit	OpenColl JN-DK Page 90
LCD 110	4 lines of 20 characters each	– 12 F keys (with LED) – Special Function Keys – Numeric keypad	Backlit	RS-422 JN-DK Page 90
LCD 12	2 lines of 16 characters each	– 4 F keys – Special Function Keys – Numeric keypad	Designed for: installation in hand-held operator consoles	OpenColl JN-DK Page 90
LCD 16	4 lines of 20 characters each	– 5 F keys (with LED)	Allows modular expansion by keyboard (NUM25) and handwheel modules (HR1)	RS-422 JN-DK Page 90
LCD 17	Graphic display 128 x 240 pixels	– 6 F keys (with LED) – Special Function Keys – Numeric keypad Cursor keypad	Visualization with: Numeric objects Text variable Bargraph D/A transfer	RS-422 JN-DK Page 90
LCD 19	Graphic display 240 x 120 pixels	– 6 F keys (with LED) – Special function keys with alpha-numeric function – Numeric keypad – Cursor keypad	Visualization with: Numeric objects Text variable Bargraph D/A transfer	RS-422 JN-DK Page 90
LCD 23	2 lines of 24 characters each	– Cursor left – Cursor right – ENTER	5 mm character height	RS-422 JN-DK Page 90
LCD 23L	1 line of 16 characters	– Cursor left – Cursor right – ENTER	8 mm character height	RS-422 JN-DK Page 90

Overview of HMIs (HMIs)				
Type	**Display**	**Keys**	**Comment**	**Interface cable**
LED 23	1 line of 8 characters		12 mm character height 7-segment LED	RS-422 JN-DK Page 90
LCD 25	2 lines of 24 characters each	– 5 F keys (with LED)	5 mm character height backlit	RS-422 JN-DK Page 90
LCD 25L	1 line of 16 characters		8 mm character height backlit	RS-422 JN-DK Page 90
LED 25	1 line of 8 characters (LED)		12 mm character height 7-segment LED	RS-422 JN-DK Page 90
LCD 27	2 lines of 24 characters each	– 5 F keys – Cursor keypad – Clear – ENTER		RS-422 JN-DK Page 90
LCD 34	2 lines of 24 characters each	– 5 F keys – Special Function Keys – Numeric keypad	Backlit	RS-422 JN-DK Page 90
LCD 52	4 lines of 16 characters each	– 6 F keys – Numeric keypad		RS-422 Page 93
LCD 54	4 lines of 16 characters each	– 8 F keys – Numeric keypad	Backlit Emergency Stop	RS-422 Page 93
LCD 54 Z	4 lines of 16 characters each	– 8 F keys – Numeric keypad	Backlit Emergency Stop "Dead man's" Key	RS-422 Page 93
LCD 60	2 lines of 40 characters each	– 8 F keys (with LED) – Numeric keypad	Backlit	RS-422 Page 91
LCD 62	2 lines of 40 characters each	– 15 F keys (with LED) – 1 Info key	Backlit	RS-422 Page 91

18.2 Description of Connections

For connecting HMIs to the LCD input of the JC-647 the cables listed in the table in chapter 18.1 "Technical Data" on page 137 are used.

18.3 Multi-Display Mode

Multi-display mode allows a JC-647 controller to be operated with up to four LCD HMIs. When doing so, the same or different texts and/or register contents can be displayed on the various HMIs.

- Specific parameters for the LCD HMI used are described in the corresponding Operator's Manual.
- An individual number must be assigned to each LCD HMI.
- If only one LCD HMI is used, value 0 is always assigned.
- If more than one LCD HMI is used, a value between 1 and 4 is assigned to each LCD HMI starting with 1. In this case, a display with # 1 must exist.

The display with number 1 is the master LCD. After power-up only the first LCD HMI is synchronized with the JC-647 controller. The other LCDs remain inactive until they receive command signals.

Note!

User input and monitor mode (which is invoked by pressing the key "R" or "I/O") can be activated at the same time only for one display.

Note!

- The voltage for several LCD HMIs cannot be supplied anymore from the controller only.
- Therefore, the LCD HMIs need to be supplied by a separate power supply with a DC voltage of 15 V to 30 V.
- Power consumption of individual LCD HMIs (approx. 300 mA) has to be taken into account when designing a system and when using it.
- For connecting several HMIs to the LCD port of the controller you need an adaptor or modified connecting cables.
- Different from the standard design, the connection cables must be modified, see Fig. 45.
- Only HMIs with RS-422 interface can be used in multi-display mode.
- Examples and hints on multidisplay mode are given in Application Note 002.

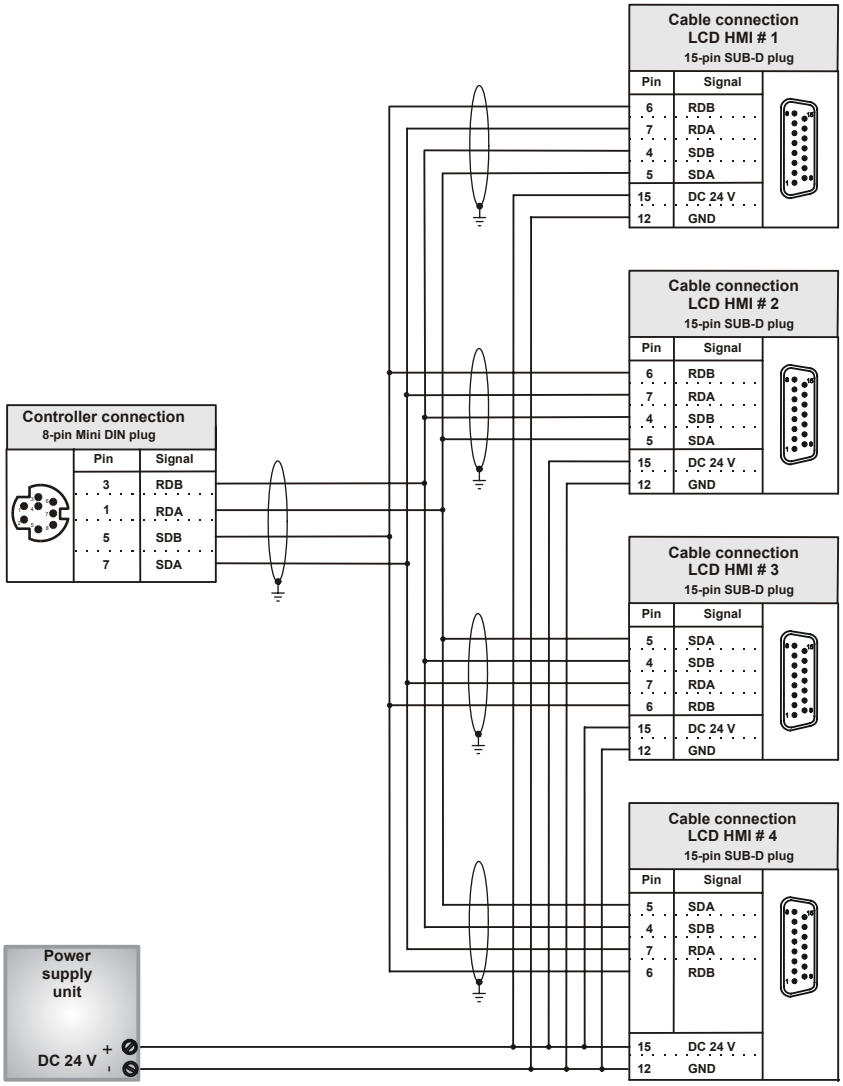


Fig. 45: Pin Assignment of the Connecting Cable for Several LCD HMIs

Important!

- When fabricating the connection cables, the following minimum requirements apply also with a view to EMC:

1.	Number of cores:	6
2.	Core cross-sectional area:	0.14 mm^2
3.	Connector (male):	SUB-D, metallized
4.	Maximum cable length:	100 m
5.	Shield:	Complete shielding, no paired shielding

- The shield must be connected to the metallised connector housings on both ends of the cable with the greatest possible surface area.

18.4 Programming the HMIs

This chapter describes the instructions necessary for programming the HMIs. For programming, the following instructions will be used:

- DISPLAY_TEXT
- DISPLAY_TEXT_2
- DISPLAY_REG / DISPLAY_VALUE
- USER_INPUT

18.4.1 Displaying Texts

The following instruction is used to display text on the HMI:

JetSym

```
DISPLAY_TEXT (<DeviceNo>, <Cursorpos>, "<Text>")
DISPLAY_TEXT_2 (<DeviceNo>, <Cursorpos>, "<Text1>", "<Text2>")
DISPLAY_TEXT (<DeviceNo>, <Cursorpos>, @<RegNr>)
```

JetSym ST

```
DISPLAY_TEXT (<DeviceNo>, <Cursorpos>, "<Text>")
DISPLAY_TEXT_2 (<DeviceNo>, <Cursorpos>, "<Text1>", "<Text2>")
DISPLAY_TEXT (<DeviceNo>, <Cursorpos>, <StringVar>);
```

Meaning of Parameters

Device Number

The parameter "Device Number" is specified by entering values from 0 through 11.

#0	The device number is directly read out of register 61787.
#1 through #4	Selection of an HMI in multi-display mode.

Cursor Position

This parameter is used to define the cursor position where the first character of the text is to appear.

If cursor position 0 is set and if the special register contains the value 0, then new text is attached to any already existing text. The cursor is located at exactly the same position where it had been positioned after execution of the last "DISPLAY_TEXT" instruction.
If the value in register 61648 is not 0, the value specified in register 61648 represents the cursor position.
For details on the function of the special register 61648, please refer to Page 291.

Cursor Positions of Various HMIs		
Type	**Cursor Positions**	
LCD 9	1. line: 2. line:	1 through 24 25 through 48
LCD 10	1. line: 2. line:	1 through 24 25 through 48
LCD 12	1. line: 2. line:	1 through 16 17 through 32
LCD 16, LCD 110	1. line: 2. line: 3. line: 4. line:	1 through 20 21 through 40 41 through 60 61 through 80
LCD 17	Status line:	1 through 40
LCD 19	Status line:	1 through 40
LCD 23	1. line: 2. line:	1 through 24 25 through 48
LCD 23L	1st line:	1 through 16
LED 23	1st line:	1 through 8
LCD 25	1. line: 2. line:	1 through 24 25 through 48
LCD 25L	1st line:	1 through 16
LED 25	1st line:	1 through 8
LCD 27	1. line: 2. line:	1 through 24 25 through 48
LCD 34	1. line: 2. line:	1 through 24 25 through 48
LCD 52, LCD 54(Z)	1. line: 2. line: 3. line: 4. line:	1 through 16 17 through 32 33 through 48 49 through 64
LCD 60	1. line: 2. line:	1 through 40 41 through 80

Text

The text to be displayed can be specified here.

Control Characters for Text Output

This applies only to devices # 0 through # 4:

After a reset, the character “$“ is defined as DELEOL by special register 61461, and as DELSCR by special register 61462 (see also Page 290).

In this case, the two characters "_" and "$" serve as control characters for text output.

The “$“ character deletes the rest of a line starting from the present cursor position. If this character is to be displayed, the character code for DELEOL can be modified in special register 61461.

When the “_“ character is used, the displayed text is first deleted and then the given text is displayed starting from cursor position 1 (irrespective of the specified parameter). This character is only useful when it is placed at the beginning of the text. Else, the first part of the text would be displayed first and would then be deleted immediately.
If this character is to be displayed, the character code for DELSCR can be modified in special register 61462.

Example 1:

JetSym	`DISPLAY_TEXT (0, 0, "_Actualposition:")`
JetSym ST	`DISPLAY_TEXT (0, 0, "_Actualposition:")`

By using this instruction the entire LCD display is deleted first, and "Actual position:" is then displayed on the upper line of the display (beginning of text when cursor position = 1). The cursor position could be any other number because this number is ignored after the DELSCR character.
The following display will appear:

Example 2:

JetSym	`DISPLAY_TEXT (0, 25, "_Set Position:$")`
JetSym ST	`DISPLAY_TEXT (0, 25, "_Set Position:$");`

After issuing this instruction, the text "Set Position:" is written at the given cursor position, i.e. at the beginning of the second line of the display. Then, the rest of this line is deleted.

Example 3:

JetSym	`DISPLAY_TEXT (0, 0, "ERROR")`
JetSym ST	`DISPLAY_TEXT (0, 0, "ERROR")`

After issuing this instruction, the text "ERROR" is written starting from the present cursor position.
By doing so, this text is simply attached to any already existing text.
If register 61648 contains a value other than 0, this value is interpreted as cursor position and the text "ERROR" is written at this position.
For details on the function of the special register 61648, please refer to Page 291.

18.4.2 Displaying Registers

A register value can be output on an HMI by the following instruction:

JetSym	`DISPLAY_REG (<DeviceNo<, <Cursorpos> <RegNo>)`
JetSym ST	`DISPLAY_VALUE (<DeviceNo>, <Cursorpos>, <VarName>);`

The parameters "DeviceNo" and "CursorPos" have exactly the same function as described for the DISPLAY_TEXT instruction.

JetSym

In addition, the number of an integer or floating point register is to be specified. Of course, this is the number of the register, the contents of which are to be displayed. Indirect addressing can be used for this purpose.

JetSym ST

Here, the name of an integer or floating point variable is to be specified as third parameter. The content of this variable is displayed. Of course, this is the number of the register, the contents of which are to be displayed. Indirect addressing using the pointer operator "@" is also allowed.

18.4.3 USER_INPUT

The USER_INPUT instruction is for reading register or variable values which have been entered via HMI.

JetSym	`USER_INPUT (<DeviceNo>, <CursorPos>, <RegNo>)`
JetSym ST	`USER_INPUT (<DeviceNo>, <CursorPos>, <VarName>);`

The same conditions apply to the two parameters "Device Number" and "Cursor Position" as to the DISPLAY_TEXT instruction. If cursor position 0 is entered, the value in register 61649 is taken as cursor position for user input. If this value is 0, which is the value of the register after a reset, then the present cursor position is used for user input.

JetSym

The register number is the number of the register to which the value, that has been entered, is to be assigned. Here, simple indirect register addressing is possible as well.

JetSym ST

Here, the name of the integer or floating point variable is specified to which the entered value is to be assigned. Here, simple indirect addressing is possible as well.

Note!

As a rule, 8 characters are available for USER_INPUT. This value, which is stored in register 61455, can be modified.

Example:

JetSym	`DISPLAY_TEXT (0, 1, "_New Position ?")` `USER_INPUT (0, 21, 100)`
JetSym ST	`DISPLAY_TEXT (0, 0, "_New Position ?")` `USER_INPUT (0, 21, rNewPos);`

To provide meaningful user guidance, the USER_INPUT instruction is usually combined with the DISPLAY_TEXT instruction.
The effect of these two instructions is that the text "New Position?" is displayed on the top left of the display. Next, the controller is waiting for a numeral to be entered. This numeral is stored in register 100 and serves as new set position for positioning purposes.

18.4.4 USER_INPUT: Default Value

An additional special register exists, i.e. register 61976, which suggests a value (default value) to the user when issuing the USER_INPUT instruction.

Instead of 0, the value in register 61976 appears on the display followed by the cursor. The operator can either confirm this value by pressing ENTER, or modify it. The modified value is confirmed by pressing ENTER.
Pressing C (clear) deletes the input and the suggested value from register 61976 appears again.

Example 1:

JetSym
```
USER_INPUT (0, 1, 100)
```

JetSym ST
```
USER_INPUT (0, 1, MEM[100]);
```

The displayed value 0 is the defined value from register 61976.

Example 2:

JetSym
```
REGISTER_LOAD (61976, 88)
USER_INPUT (0, 1, 100)
```

JetSym ST
```
MEM[2815] := 88;
USER_INPUT (0, 1, MEM[100]);
```

The displayed value 88 is the defined value from register 61976.

19 Submodules JX6-SB, and JX6-SB-I

In operating mode "Master/Slave", the JX6-SB(-I) submodule is a CAN interface module for connection to the system bus used by Jetter AG. This interface module serves for connecting JX2-I/O, JX-SIO, Lumber LJXz-CSL-... modules, as well as third-party modules to the JetControl 647. In addition, up to eight JX2-Slave modules, JetMove 2xx and JetMove 6xx modules can be connected to a JX6-SB-I.
It also enables networking of several controllers of the JetControl 647 and DELTA family in "Master/Master" mode.

Important!

This chapter gives a description of functions offered by the modules JX6-SB and JX6-SB-I with software release 2.12.

For more information refer to the User Information on the JX6-SB / JX6-SB-I module.

19.1 Technical Data

Design	
Dimensions (H x W x D in mm)	17.0 x 54.5 x 120.0
Weight	60 g
Installation	To be plugged into socket # 1 through 3 of the JC-647 (see Fig. 4)

Electrical Data	
Power supply	+5 V -4 % / +4 %
Power consumption	approx. 2 W
Baud rate of JETTER System Bus	125 kBaud up to 1 MBaud
Maximum cable length of the JETTER system bus (between controller and distributed expansion modules)	200 m at 125 kBaud 30 m at 1 MBaud

Functional Data	
Design of the JETTER System Bus	A CAN bus which has been modified to meet JETTER's software and hardware requirements.
Can be started in 3 modes:	• Master-Slave D-CAN2 • Master-Slave JX6-SB • Master-Master

19.1.1 Operating Mode "Master-Slave JX6-SB"

Number of modules connected to the bus	
Operating Mode "Master-Slave JX6-SB"	
Maximum amount of JX2-I/O Modules	31
Maximum amount of JX2-Slave modules (JX6-SB-I required)	8
Maximum number of JX-SIO modules or third-party modules (limited by the maximum amount of IOs of the JX6-SB module)	10
Total amount of IOs	496

Important!

When designing and configuring the JETTER system bus, attention must be paid not to exceed the maximum amount of IOs.

Third-party Modules Connected to the System Bus	
Manufacturer	**Product Designation**
Bürkert GmbH & Co. KG	Valve terminal type 8640
Festo AG & Co.	Valve Terminals: CPV10-GE-CO2-8 CPV14-GE-CO2-8 CPV18-GE-CO2-8 CPX-FB14
SMC Pneumatik GmbH	SI-Unit: • EX120 - SCA1 • EX121 - SCA1 • EX122 - SCA1 • EX250-SCA1
Jenaer Antriebstechnik	Ecostep Drives: • 100-AA-000 • 100-LA-000 • 100-PA-000 • 200-AA-000 • 200-PA-000 • 200-ZA-000 • 200-QA-000 • 216-AA-000 • 216-PA-000 • 216-ZA-000 • 216-QA-000
Werner Riester GmbH & Co. KG	Milan Drives: • MI 1.5/075 • MI 2/090 • MI 4/110
Lenze GmbH & Co. KG	Special Fieldbus Function Module, Type 2175, SW Version 1.0 Supported basic devices: • 8201 - 8204 • 8211 - 8218 • 8221 - 8227 • 8241 - 8246 • 8200 vector • 8200 vector, Cold plate
Vacon	Vacon NX Frequency converter

Please read the corresponding User Informations for connecting and operating third-party modules.

Important!

In order to ensure flawless functioning of the remote arrangement, the following instructions as regards the system bus topology must be met.
Failure to meet these instructions will result in malfunctions of individual modules or breakdown of the entire system configuration.

Instructions on system bus topology

- A maximum of 31 JX2-I/O modules, the JX2-PS1 modules not being counted, can be connected.
- For each remote module group, at least one JX2-PS1 module or one JX2-IO16 module will be needed. Even with intelligent modules, the JX2-PS1 modules must be located at the beginning of the JX2-Slave module set, in order to reduce electromagnetic interferences.
- One JX2-PS1 module is to supply 5 JX2-I/O modules as a maximum.
- One JX2-IO16 module is to supply 3 JX2-I/O modules as a maximum.
- Up to eight JX2-Slave modules, JetMove 2xx and JetMove 6xx modules can be connected to a JX6-SB-I.
- Power supply of JX2-Slave modules (such as JX2-SV1, JX2-PID1, JX2-SM1D, etc.) is provided by an individual power supply unit (DC 24 V), and not by a JX2-PS1 module.
- JX2-Slave and JetMove modules are not capable of supplying JX2-I/O modules with power.
- JX-SIO modules are supplied with DC 24 V by a PSU of their own. Therefore, they do not require a JX2-PS1 module.
- The last JX-SIO module has to be equipped with a terminating resistor.
- The maximum line length and tap line length depending on the baud rate must not be exceeded.

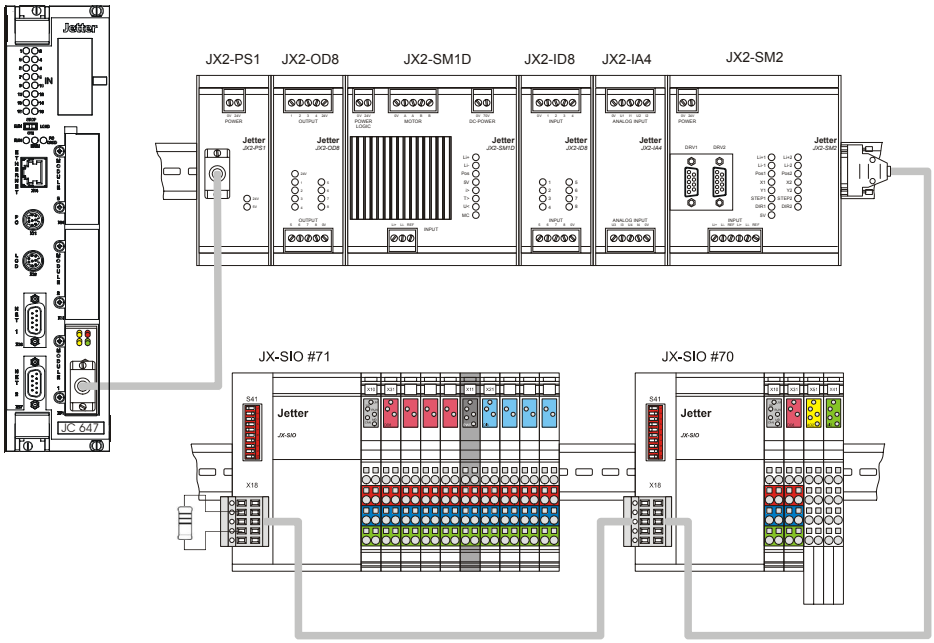


Fig. 46: Remote Arrangement on the JETTER System Bus

Assigning System Bus Module Numbers

When assigning system bus module numbers, JX2-I/O modules, JX2-Slave modules and JX-SIO modules have to be counted separately. In this connection, the following rules apply:

- All JX2-I/O modules are counted up, beginning from the JX6-SB(-I) module. When doing so, all JX2-Slave modules, JX-SIO and JX2-PS1 modules are being ignored.
- The number "2" is assigned to the first JX2-I/O module connected to the JX6-SB(-I) submodule.
- All JX2-Slave modules are counted up, beginning from the JX6-SB-I submodule. When doing so, all JX2-I/O modules, JX-SIO and JX2-PS1 modules are being ignored.
- The number "2" is assigned to the first JX2-Slave module connected to the JX6-SB-I submodule.
- To all JX-SIO modules the module number set by switch S41 is assigned, irrespective of their position on the system bus.

The module numbers for system bus configuration as given in Fig. 46 are assigned as follows:

System Bus Module Numbers		
Module	**IO Module Number**	**Slave Module Number**
JX6-SB-I	-	-
JX2-PS1:	-	-
JX2-OD8	2	-
JX2-SM1D	-	2
JX2-ID8	3	-
JX2-IA4	4	-
JX2-SM2	-	3
JX-SIO	70	-
JX-SIO	79	-

19.1.2 Operating Mode "Master-Master"

Number of modules connected to the bus	
Operating Mode "Master-Master"	
Maximum amount of JX6-SB submodules	12
Maximum amount of register data per JX6-SB submodule	64 registers
Internal receiving buffer per JX6-SB submodule	12 x 2 x 64 registers 12 devices x 2 data buffers x 64 registers

Among the ports, 2 port areas are of particular importance in "Master/Master".

Port Area	Register Type
1	Configuration and Status Registers
2	Register Data

Bus Terminating Resistor	
The bus terminating resistor can be activated or deactivated through a command. The bus terminating resistor at both ends of the system bus must be activated in the JX6-SB submodules. For all other devices it must be deactivated.	
Deactivating the bus terminating resistor	Command 13 (register 11m101)
Activating the bus terminating resistor	Command 14 (register 11m101)

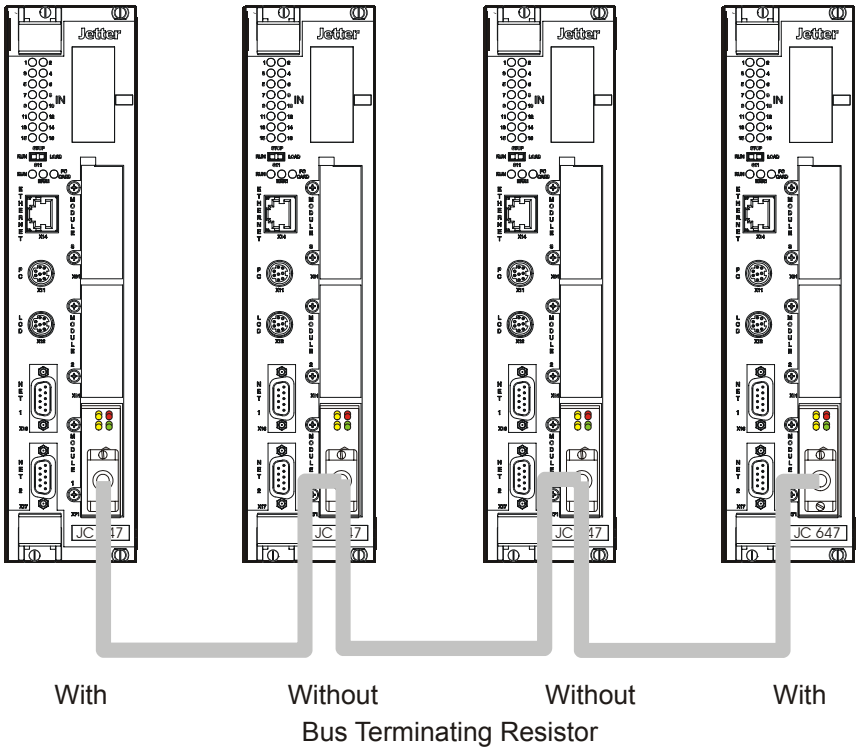


Fig. 47: Networking of several JC-647 controllers in Master/Master mode

Example: Command Register of the JX6-SB Submodule

A command is to be entered into register 1 of port 1 (command register). The JX6-SB submodule is located in socket # 2 on the JC-647.

Initializing the Master/Master mode of the JX6-SB submodule

```
REGISTER_LOAD (112101, 10)
```

Deactivating the bus terminating resistor

```
REGISTER_LOAD (112101, 13)
```

19.2 Operating Modes

The JX6-SB submodule can be operated in three different modes:
- Master/Slave D-CAN2
- Master/Slave JX6-SB
- Master/Master

Operating Modes - Overview			
Function	**Master/Slave D-CAN2**	**Master/Slave JX6-SB**	**Master/Master**
From software release ...	V1.00	V2.10	V1.10
Initialization command (register 11m101)	1	30	10
Expansion Modules			
Add. JX6-SB modules			✓
JX2-I/O modules	✓	✓	
JX2-Slave modules (only JX6-SB-I)	✓	✓	
JX-SIO		✓	
LJX7-CSL-... Modules		✓	
FESTO CP-FB modules	✓		
Third-party modules (refer to Page 149)		✓	
IO Numbering and Register Numbers			
IO numbers of JX2-I/O modules	1m001 .. 1m999[1]	$m_1$0201 .. $m_1$3216[2]	
IO numbers of JX-SIO modules, IO Numbers of LJX7-CSL-... modules		$m_1$7001 .. $m_1$7964	
Registers of JX2-I/O modules	3m03000 .. 3m03309	3m03000 .. 3m03309	
Registers of JX-SIO modules, registers of LJX7-CSL-... modules		3m07001 .. 3m07964	
Registers JX2-Slave modules (only JX6-SB-I modules)	3m12100 .. 3m19999	3m12100 .. 3m19999	

[1] m = Submodule socket (1 .. 3)
[2] m_1 = Submodule socket + 1 (2 .. 4)

19.3 System Bus - Baud Rate

The system bus of Jetter AG can be operated at baud rates between 125 kBaud and 1 MBaud.

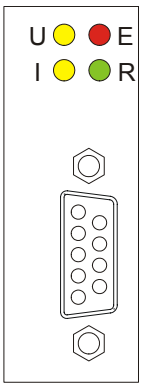
Generally, the max. permissible line length of the system bus becomes shorter with increasing baud rate. At the same time, however, the data transmission speed on the system bus increases with increasing baud rate.
Therefore, for every individual application you have to decide whether the system bus should be operated with the maximum transfer rate or the maximum cable length.

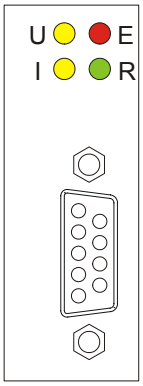
The permissible baud rates of the system bus also depend on the modules connected to it.

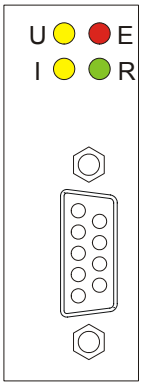
Permissible Baud Rates					
JX2-I/O Modules JX2-Slave Modules	**JX-SIO LJX7-CSL-...**	**125 kBaud**	**250 kBaud**	**505 kBaud**	**1000 kBaud**
✓		✓	✓	✓	✓
	✓	✓	✓	✓	✓
✓	✓	✓			✓

19.4 Diagnostic Indicators (LEDs)

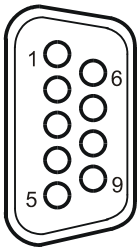
Four LEDs are located in the front panel of the JX6-SB(-I) submodule which indicate the various modes the submodule is in.

Description of LEDs			
After power-up			
View	**LED**	**Status**	**Meaning**
	'R' (green)	Slowly blinking	The JX6-SB submodule is ready for initialization commands.
	'R' (green)	Blinking twice, then long interval	The JX6-SB-I submodule is ready for initialization commands.
	'R' (green) 'E' (red) 'I' (yellow)	Blinking fast	The JX6-SB(-I) submodule has no valid firmware.

Description of LEDs			
Master/Slave Mode			
View	**LED**	**Status**	**Meaning**
	'R' (green)	is lit	This LED indicates successful initialization (command 1 or command 30) of the JX6-SB(-I) submodule.
	'E' (red)	is lit	When the operating system firmware of the JX6-SB(-I) submodule is running, the red LED indicates the state of the error bits in the status register (11m100). When an error has occurred, the LED "E" is lit.
	'I' (yellow)	is lit	The yellow LED will remain lit, as long as the initialization of the system bus is active.
	'U' (yellow)	is lit	This yellow status LED is used for indicating the scan cycle of all input modules. After each reading of the input conditions of all connected modules the LED changes its state. Thus, the blinking frequency indicates the duration of the input module updating process.

Description of LEDs			
Master/Master Mode			
View	**LED**	**Status**	**Meaning**
	'R' (green)	is lit	This LED indicates successful initialization (command 10) of the connected JX6-SB(-I) submodule.
	'E' (red)	is lit	A system bus error has occurred.
	'I' (yellow)	is lit	An accumulation of errors has been detected on the system bus (warning limit has been reached).
	'U' (yellow)	is lit	An overflow is reported by at least one receiving buffer.

19.5 Description of Connections

Pin Assignment - Female SUB-D connector, 9 pins, JX6-SB(-I)			
View	**Pin**	**Signal**	**Comment**
	1	CMODE0	[1]
	2	CL	
	3	GND	
	4	CMODE1	[1]
	5	TERM	Unused [1]
	6	-	Unassigned
	7	CH	
	8	-	Unassigned
	9	-	Do not connect

[1] will not be required for master-master operation

19.5.1 JETTER System Bus Cable Specification

Specification of Connectors

On the JX6-SB(-I) submodule side
- 9-pin male SUB-D connector in metallized housing
- Diameter of the cable apt for connecting: 0.25 - 0.60 mm^2

On the opposite side of the cable
- 9-pin female SUB-D connector in metallized housing
- Diameter of the cable apt for connecting: 0.25 - 0.60 mm^2

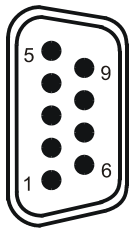
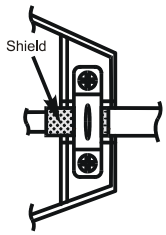
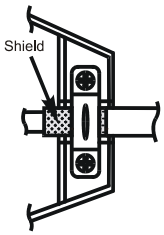
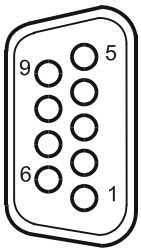
System Bus Cable Specification

• Number of cores	5	
• Core cross-sectional area	0.25 - 0.34 mm^2	at 1 MBaud
	0.34 - 0.50 mm^2	at 500 kBaud
	0.34 - 0.60 mm^2	at 250 kBaud
	0.50 - 0.60 mm^2	at 125 kBaud
• Max. cable capacitance	60 pF/m	
• Maximum resistivity	70 Ω/km	at 1 MBaud
	60 Ω/km	at 500 kBaud
	60 Ω/km	at 250 kBaud
	60 Ω/km	at 125 kBaud
• Maximum cable length [1]	30 m	at 1 MBaud
	100 m	at 500 kBaud
	200 m	at 250 kBaud
	200 m	at 125 kBaud
• Max. tap line length	0.3 m	at 1 MBaud
	1 m	at 500 kBaud
	3 m	at 250 kBaud
	-	at 125 kBaud
• Max. overall tap line length	3 m	at 1 MBaud
	39 m	at 500 kBaud
	78 m	at 250 kBaud
	-	at 125 kBaud

[1] The following rule of thumb applies to the system bus cable: Each JX2-I/O module connected to the bus reduces the maximum cable length by approx. 1 m.

Cable Shielding

- Complete shielding, no paired shielding
- The shield must be connected to the connector housings on both ends of the cable with the greatest possible surface area.
- If there are long lines between two modules on the system bus, the line shield must be connected with functional earth (FE) approximately every 10 m because of EMI precautions.

JETTER System Bus Cable		
	Shield	
	 	
BUS-OUT	Connect shield with the greatest possible surface area! Use metallized housing only!	**BUS-IN**
Pin	**Signal**	**Pin**
1	CMODE0	1
2	CL	2
3	GND	3
4	CMODE1	4
5	TERM (not connected)	5
6	Unassigned	6
7	CH	7
8	Unassigned	8
9	Do not connect	9

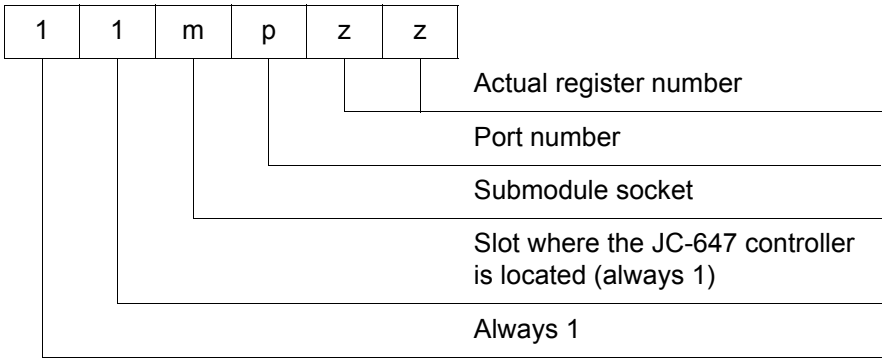
19.6 Firmware - Master-Slave JX6-SB(-I)

19.6.1 Overview of Registers

Register Areas - Master-Slave JX6-SB	
Register Area	**Description**
11m100 .. 11m163	Configuration and status registers of the JX6-SB(-I) submodule This area is for commissioning the JX6-SB submodule, reading out the software version, etc.
3m0 2000 .. 3m0 2999	System Bus Special Registers This area is for diagnosing the JX6-SB submodule, for detecting timeouts, and short circuits, for adjusting the baud rate, for configuring dummy modules, for reading out the modules currently connected to the system bus, etc.
3m0 3000 .. 3m0 3299	Registers of JX2-I/O modules With the help of these registers, analog values of JX2-I/O modules can be read and written, etc.
3m0 4000 .. 3m0 4999	Register overlay of JX2-I/O modules
3m0 5000 .. 3m0 5999	Overlaid IO registers for digital inputs and registers for digital and analog input data of JX-SIO modules
3m0 6000 .. 3m0 6999	Overlaid IO registers for digital inputs and registers for digital and analog output data of JX-SIO modules
3m0 7000 .. 3m0 7999	Configuration and status registers of JX-SIO, Lumberg LJX7-CSL and third-party modules
3m 12100 .. 3m 19999	Configuration and status registers of the JX2-Slave and JetMove modules

Configuration and status registers of the JX6-SB(-I) submodule

By way of example **REG 1xyzzz** will be demonstrated how the registers are numbered.



The registers are addressed with the help of a 6 digit number:

- The first digit is always **1**.
- The second digit is always **1**.
  The second digit specifies the **slot**, where the JetControl 647 controller module is located.
- The third digit **m** specifies the JetControl 647 **submodule socket** where the JX6-SB-I submodule is located..
  m = Submodule socket (1 .. 3).
- The fourth digit **p** specifies the **port number**. The port number "p" is to distinguish the four register ranges on the JX6-SB(-I) submodule.
  p = Port number (1..4).
- The digits five and six **zz** specifiy the actual **register number** with the letters zz corresponding to the register numbers from 0 to 63.

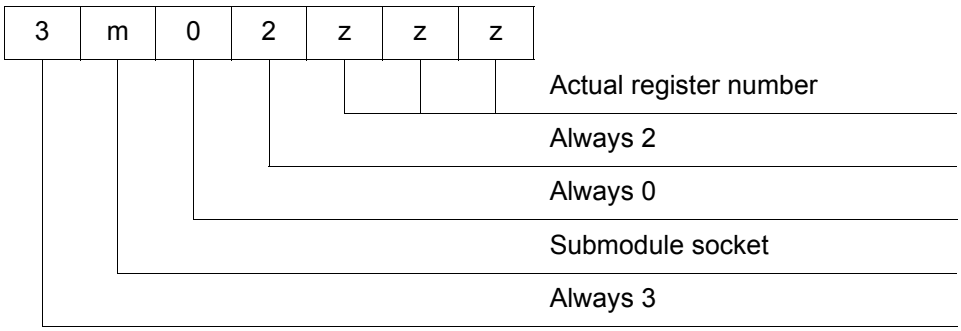
Example: Command Register of the JX6-SB Submodule

Command 30 is to be entered into register 1 of port 1 (command register). The JX6-SB(-I) submodule is located in socket # 2 on the JC-647.

```
REGISTER_LOAD (112101, 30)
```

System Bus Special Registers

By way of example **REG 3m02zzz** will be demonstrated how the registers are numbered.

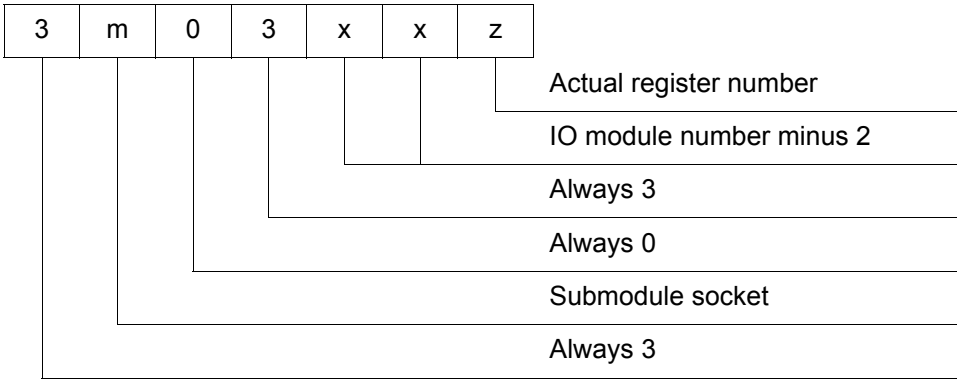


The registers are addressed with the help of a 7 digit number:

- The registers are addressed with the help of a 7 digit number.
- The first digit is always **3**.
- The second digit **m** specifies the JetControl 647 **submodule socket** where the JX6-SB-I submodule is located.
  m = Submodule socket (1 .. 3).
- The third digit is always **0**.
- The fourth digit is always **2**.
- The digits five, six and seven **zzz** specify the actual **register number**. One of the 100 possible registers will be selected by the register number "zzz".

Register Area for JX2-I/O Modules

By way of example **REG 3m03xxz** will be demonstrated how the registers are numbered.

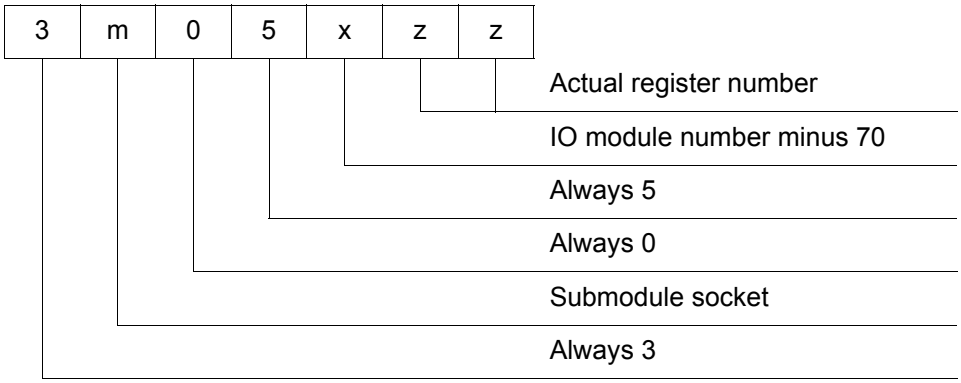


The registers are addressed with the help of a 7 digit number:

- The first digit is always **3**.
- The second digit **m** specifies the JetControl 647 **submodule socket** where the JX6-SB-I submodule is located.
  m = Submodule socket (1 .. 3).
- The third digit is always **0**.
- The fourth digit is always **3**.
- The digits five and six **xx** specify the **IO module number minus 2** (0.. 30).
- The digit seven **z** specifies the actual **register number**. One of the 10 possible registers will be selected using this register number.

Register area for JX-SIO, Lumberg LJX7-CSL-..- and third-party modules

By way of example **REG 3m05xzz** will be demonstrated how the registers are numbered.

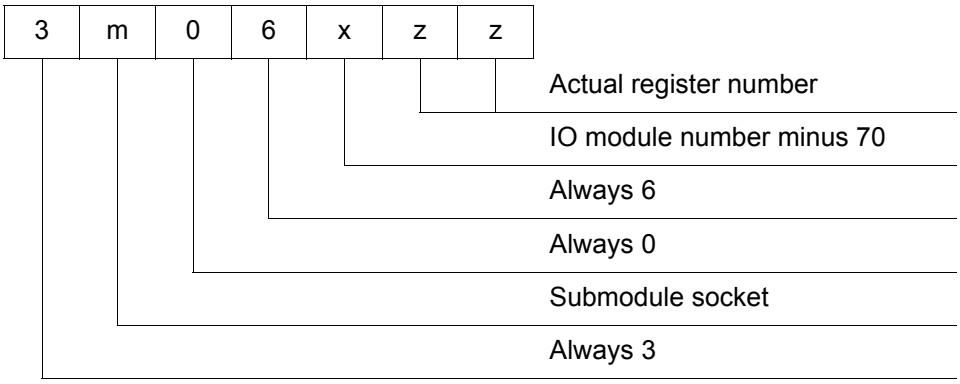


The registers are addressed with the help of a 7 digit number:

- The first digit is always **3**.
- The second digit **m** specifies the JetControl 647 **submodule socket** where the JX6-SB-I submodule is located.
  m = Submodule socket (1 .. 3).
- The third digit is always **0**.
- The fourth digit is always **5**.
- The digits five **x** specifies the **IO module number minus 70** (0.. 9).
- The digits six and seven **zz** specify the actual **register number**. One of the 100 possible registers will be selected using this register number.

Registers for digital and analog output data

By way of example **REG 3m06xzz** will be demonstrated how the registers are numbered.

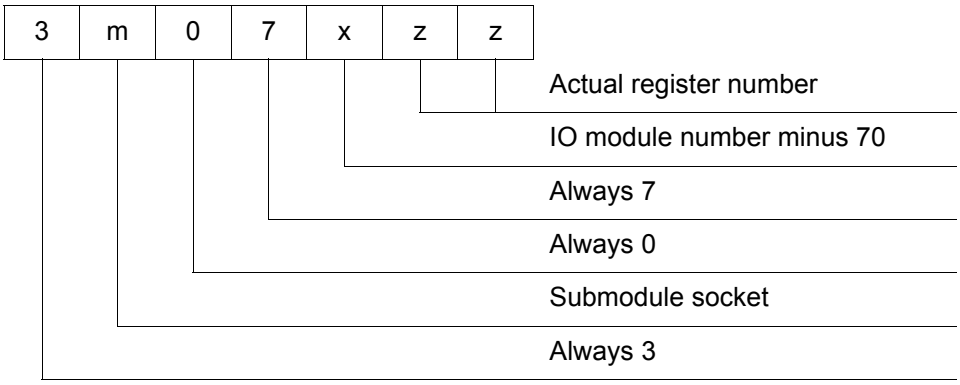


The registers are addressed with the help of a 7 digit number:

- The first digit is always **3**.
- The second digit **m** specifies the JetControl 647 **submodule socket** where the JX6-SB-I submodule is located.
  m = Submodule socket (1 .. 3).
- The third digit is always **0**.
- The fourth digit is always **6**.
- The digits five **x** specifies the **IO module number minus 70** (0.. 9).
- The digits six and seven **zz** specify the actual **register number**. One of the 100 possible registers will be selected using this register number.

Configuration and diagnostic registers

By way of example **REG 3m07xzz** will be demonstrated how the registers are numbered.

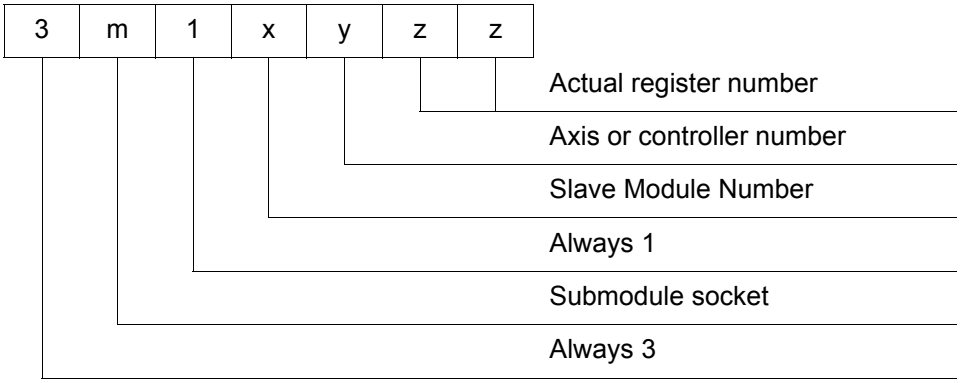


The registers are addressed with the help of a 7 digit number:

- The first digit is always **3**.
- The second digit **m** specifies the JetControl 647 **submodule socket** where the JX6-SB-I submodule is located.
  m = Submodule socket (1 .. 3).
- The third digit is always **0**.
- The fourth digit is always **7**.
- The digits five **x** specifies the **IO module number minus 70** (0.. 9).
- The digits six and seven **zz** specify the actual **register number**. One of the 100 possible registers will be selected using this register number.

Register Area for JX2-Slave Modules

By way of example **REG 3m1xyzz** will be demonstrated how the registers are numbered.

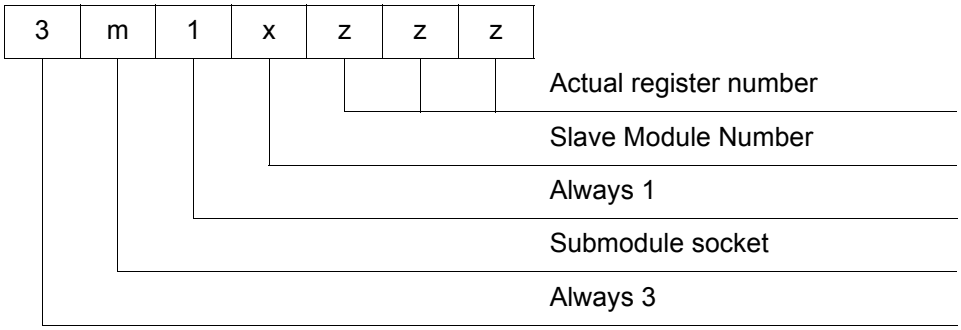


The registers are addressed with the help of a 7 digit number:

- The first digit is always **3**.
- The second digit **m** specifies the JetControl 647 **submodule socket** where the JX6-SB-I submodule is located..
  m = Submodule socket (1 .. 3).
- The third digit is always **1**.
- The fourth digit **x** specifies the **number of the slave module** connected to the system bus:
  x = Slave module number (2 .. 9).
- The fifth digit **y** is the **number of the axis or controller** (0 .. 9).
- The digits six and seven **zz** specify the actual **register number**. One of the 100 possible registers will be selected using this register number.

Register Area for JetMove 2xx and JetMove 6xx

By way of example **REG 3m1xzzz** will be demonstrated how the registers are numbered.



The registers are addressed with the help of a 7 digit number:

- The first digit is always **3**.
- The second digit **m** specifies the JetControl 647 **submodule socket** where the JX6-SB-I submodule is located..
  m = Submodule socket (1 .. 3).
- The third digit is always **1**.
- The fourth digit **x** specifies the **number of the slave module** connected to the system bus:
  x = Slave module number (2 .. 9).
- The digits five, six and seven **zzz** specify the actual **register number**. One of the 1000 possible registers will be selected using this register number.

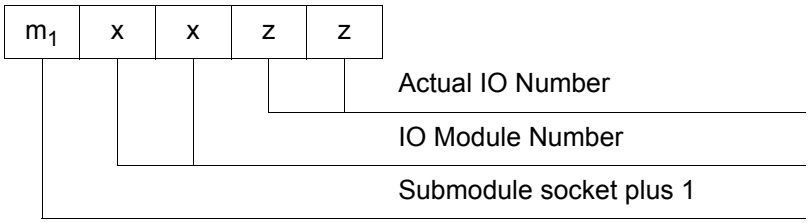
19.6.2 IO Range

In the IO range, the transferred inputs and outputs are combined.
After system bus initialization, 16 inputs and outputs are assigned to each recognized JX2-I/O module, resp. 64 inputs and outputs to each recognized JX-SIO module.
All inputs and outputs can either be accessed individually via IO instructions of the application program or with the help of the SYMPAS setup screen, or else with the help of register overlay.

Access to Inputs and Outputs using IO Instructions

Access to inputs and outputs of system bus modules is possible, in the usual way, through IO instructions of the JetSym programming language. Distinction between system bus IOs and local IOs is made by numbering of inputs, and outputs.

By way of example **IN $m_1$xxzz** or **OUT $m_1$xxzz** will be demonstrated how inputs and outputs are numbered.



The inputs and outputs are addressed with the help of five-digit numbers:

- The first digit **m_1** specifies the JetControl 647 **submodule socket** where the JX6-SB-I submodule is located **plus 1**:
  m_1 = Submodule socket + 1(2 .. 4).

- The second and third digit **xx** specifies the **number of the IO module** connected to the system bus:
  xx = IO module number (JX2-I/O): 2 .. 32, JX-SIO and LJX7-CSL-..: 70 .. 79).

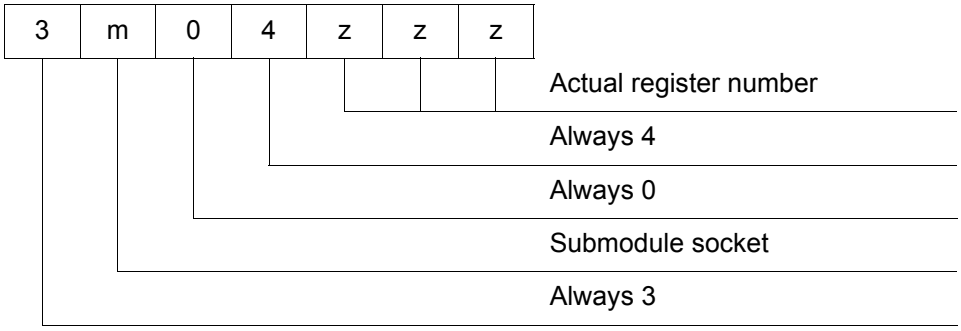
- The digits four and five **zz** specify the actual **IO number**
  (JX2-I/O: 1 .. 16, JX-SIO: 1 .. 64, LJX7-CSL-..: 1 .. 16).

Access to Inputs and Outputs by means of Register Overlaying

Besides individual access to digital inputs and outputs, several inputs and outputs can jointly be accessed by means of register overlaying. This approach allows, for example, to simultaneously delete several outputs using a single REG_ZERO instruction.

IO register overlay of JX2-I/O modules

By way of example **REG 3m04zzz** will be demonstrated how the registers are numbered.



The registers are addressed with the help of a 7 digit number:

- The first digit is always **3**.
- The second digit **m** specifies the JetControl 647 **submodule socket** where the JX6-SB-I submodule is located.
  m = Submodule socket (1 .. 3).
- The third digit is always **0**.
- The fourth digit is always **4**.
- The digits five, six and seven **zzz** specify the actual **register number**.

20 Network Operation JetWay

20.1 JETWay-H: JETTER Data Highway

The data highway JETWay-H enables several networked control systems of the JetControl, DELTA or NANO families to be controlled by a host computer. Theoretically, the max. number of nodes to be controlled is 126. However, reasonable communication on the network gets next to impossible with such a number of nodes since the transmission rate slows down. Therefore, it is advisable to limit the number of nodes to be controlled to 98. In detail, network operation means:

- Visualization
- Programming
- Data transfer
- Acquisition of production data
- Service functions; access to each control system

Note!

For a description of the wiring and parameter assignment of JETWay-H, please refer to chapter 15.4 "Programming Interface via JETWay-H" on page 94.

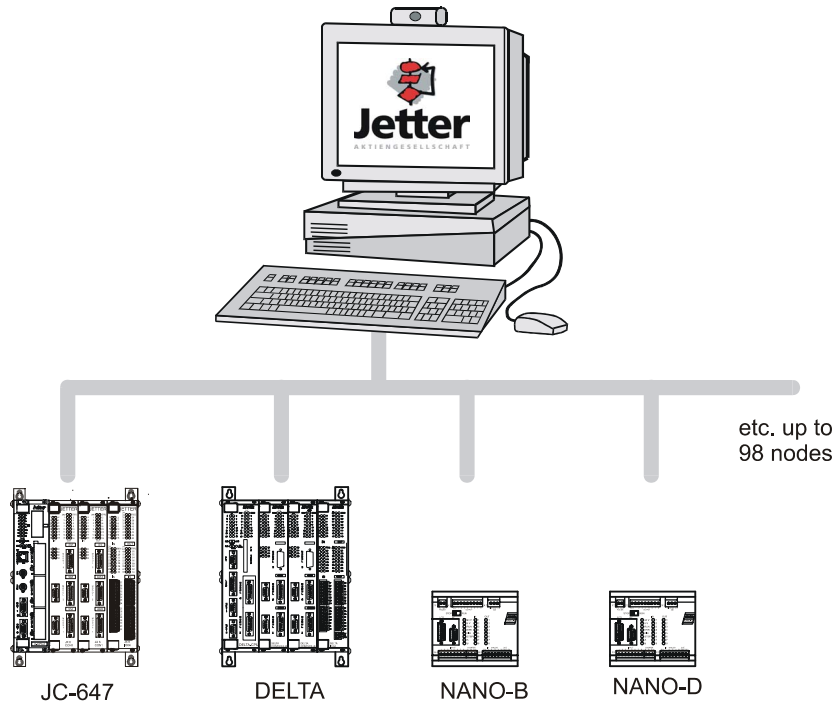


Fig. 48: JETWay-H for Management Level

20.2 JETWay-R: Process Level

The JETWay-R network has two functions:

- Hierarchical networking of DELTA and NANO control systems.
- Connection of decentralized peripheral devices such as smart IOs, remote IOs, or valve terminals.

The maximum number of nodes per level is 99. This network is a monomaster network. This means that there is one master and a maximum of 98 slaves per hierarchical level.

Note!

The controller is physically connected to the JETWay-R as slave controller via network 1 (NET 1).

If the controller is connected to the JETWay-R network as master and as slave in two different hierarchical levels, then the master is physically connected to network 2 (NET 2).

Note!

For a description of the wiring and parameter assignment of JETWay-R, please refer to chapter 15.5 "Network interface (JETWay-R)" on page 98.

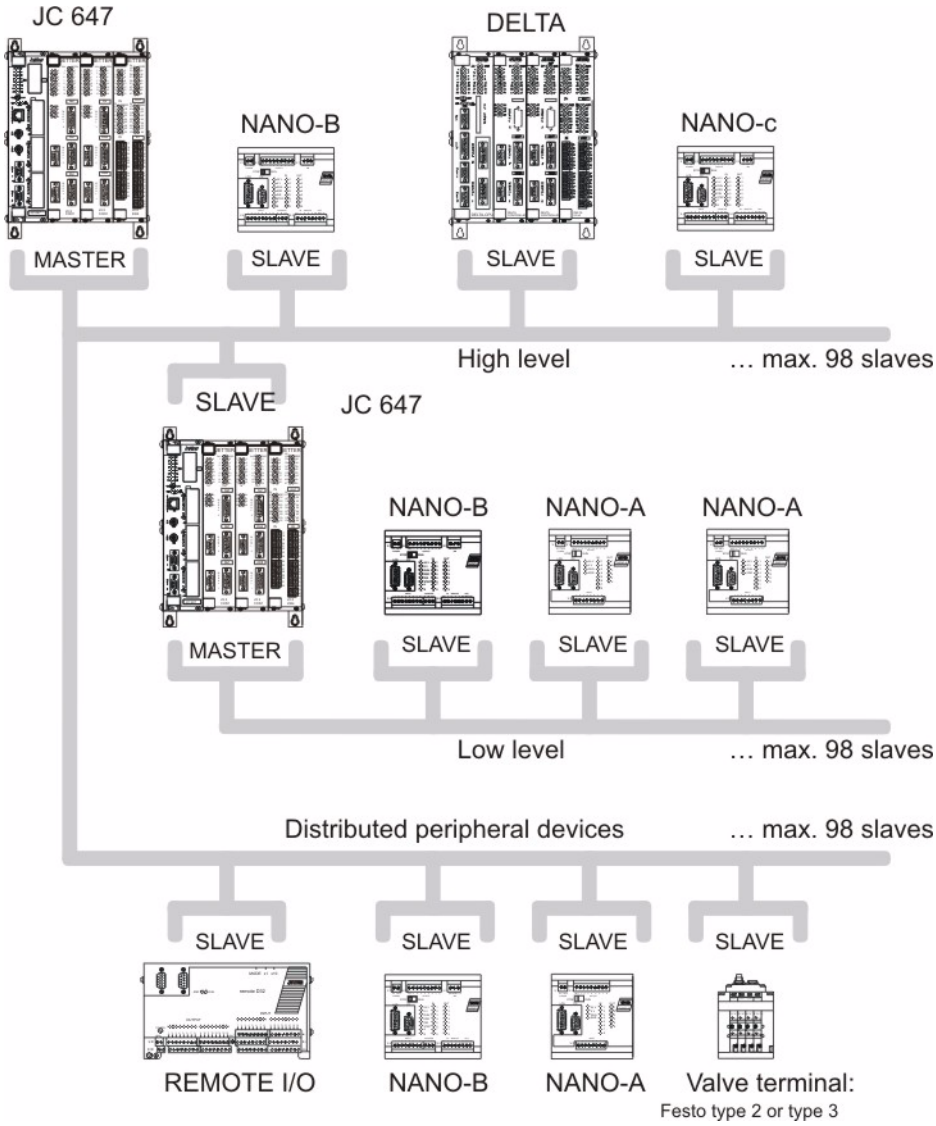


Fig. 49: JETWay-R for Process Level

20.3 N_SEND_REGISTER and N_GET_REGISTER

Note!

These register numbers are not influenced by the number offset defined in special register 61831.

In a JETWay-R network, the controller can be operated as master or as slave.

20.3.1 N_SEND_REGISTER and N_GET_REGISTER

The master controller can write values into registers of slave controllers by using the following instruction:

JetSym	`N_SEND_REGISTER (To <Network No> from <SourceReg No> to <DestReg No>)`
JetSym ST	`N_SEND_VALUE (To <Network No> from <SourceReg No> to <DestReg No>)`

- **`<Network No>: Network No`** stands for the network number of the slave controller to be addressed via the network.
- **`<Source Reg>:`** Here, the number of the register is specified, the value of which is to be transmitted to a slave via the network.
- **`<Destination Reg>:`** Here, the number of the register is specified into which the value from the master controller is to be transmitted. This register is located on the slave controller with the slave number **`Network No`**.

Example:

```
N_SEND_REGISTER (3, 100, 200)
```

Following this instruction, the value in register 100 of the master controller is entered into register 200 of the slave controller with the network number 3.

20.3.2 N_GET_REGISTER

By using the following instruction, the master controller can write values into registers of slave controllers:

JetSym	`N_GET_REGISTER (From <Network No> from <SourceReg No> to <DestReg No>)`
JetSym ST	`N_GET_VALUE (From <Network No> from <SourceReg No> to <DestReg No>)`

- **`<Network No>: Network No`** stands for the network number of the slave controller to be addressed via the network.
- **`<Source Reg>:`** Here, the number of the register is specified from which the value is to be transmitted to the master controller. This register is located on the slave controller with the slave number **`Network No`**.
- **`<Destination Reg>:`** Here, the number of the master controller register is specified into which the value from the slave controller is to be transmitted.

Example:

```
N_GET_REGISTER (3, 200, 100)
```

Following this instruction, the value in register 200 of the slave controller with the network number 3 is entered into register 100 of the master controller.

20.4 Network Operation with 50000-Numbers

Important!

Network operation with numbers in the 50000 range is not supported by JC-647.

21 Network Operation via Ethernet

Apart from the serial interfaces PC, LCD, NET1 and NET2, the JC-647 controller is equipped with an ETHERNET interface.

21.1 Setting the IP-Address

The JC-647 is delivered with the following IP-address: **192.168.1.1**

Note!

When changing the IP-address, always all the four registers must be written.

Important!

Once the values have been entered into the registers 63816 through 63819, the IP address is accepted immediately.

It is not recommended to change the IP address via the Ethernet network, since the controller cannot be accessed during this process. To change register values use one of the serial interfaces, e.g. the PC or LCD interface.

Register 63816: IP-address LSB	
Function	**Description**
Read	Current IP-address LSB
Write	New IP address LSB
Value range	0 through 255
Factory setting	1

Register 63817: IP address 2SB	
Function	**Description**
Read	Current IP address 2SB
Write	New IP address 2SB
Value range	0 through 255
Factory setting	1

Register 63818: IP address 3SB	
Function	**Description**
Read	Current IP address 3SB
Write	New IP-address 3SB
Value range	0 through 255
Factory setting	168

Register 63819: IP address MSB	
Function	**Description**
Read	Current IP-address MSB
Write	New IP address MSB
Value range	0 through 255
Factory setting	192

21.2 N_COPY_TO, and N_COPY_FROM

The content of a register can be copied from and to a JetControl controller via the Ethernet network using the instructions N_COPY_TO and N_COPY_FROM.

Since Ethernet basically offers multi-master capability, all tasks of all control systems can send such network instructions. While doing so, there is no need to apply or wait for an access authorization.

21.2.1 N_COPY_TO

The following instruction serves to copy block-by-block the content of local registers to registers of another controller.

N_COPY_TO (<IP Address>, <Source Reg>, <Destination Reg>, <Number of Registers>, <Port No>, <Mode>)

- **<IP-address>: IP Address** stands for the IP address of the controller to be addressed via the Ethernet network.
- **<Source Reg>:** Here, the number of the first register is specified whose value is to be transmitted to a controller via the network which is to be selected through **<Port No>**.
- **<Destination Reg>:** Here, the number of the register is specified into which the content of the register specified by **<Source Reg>** is to be transmitted. This register is located on the controller which is addressed by this network instruction.
- **<Number of Registers>:** This parameter specifies the number of registers to be copied block-by-block.
- **<Port No>:** At the moment, only port number 3 is available, that is, the Ethernet network card (JetIP).
- **<Mode>:** The default mode is 1. A detailed description will be given below.

The register numbers for source and destination registers can be specified directly, indirectly or double indirectly.
The value range of the parameters for the N_COPY_TO instruction is between 0 and 4,294,967,296.

Mode

The following modes are possible:

Mode	Meaning
The source register is directly addressed	
0	With **Number of registers > 1** the **Source Reg** remains constant (see example 1).
1	With **Number of registers > 1** the **Source Reg** will be incremented by one after each copying process (see example 2).
2	With **Number of registers > 1** the **Source Reg** will be decremented by one after each copying process (see example 3).

Mode	Meaning
The source register is indirectly addressed	
4	With **`Number of registers > 1`** the **`Source Reg`** remains constant.
5	With **`Number of registers > 1`** the **`Source Reg`** will be incremented by one after each copying process (see example 4).
6	With **`Number of registers > 1`** the **`Source Reg`** will be decremented by one after each copying process.

Example 1:

```
N_COPY_TO (192.168.1.20, 100, 500, 3, 3, 0)
```

The content of local register 100 is copied to registers 500 through 502 of the controller with the IP-address 192.168.1.20.

Source Reg	Contents	Implementation	Destination Reg	Contents
100	77	-->	500	77
100	77	-->	501	77
100	77	-->	502	77

Example 2:

```
N_COPY_TO (192.168.1.20, 100, 1000, 5, 3, 1)
```

The content of local registers 100 through 104 is copied to registers 1000 through 1004 of the controller with the IP-address 192.168.1.20.

Source Reg	Contents	Implementation	Destination Reg	Contents
100	77	-->	1000	77
101	3198	-->	1001	3198
102	791	-->	1002	791
103	86320	-->	1003	86320
104	13629	-->	1004	13629

Example 3:

```
N_COPY_TO (192.168.1.20, 100, 1000, 5, 3, 2)
```

The content of local registers 96 through 100 is copied to registers 996 through 1000 of the controller with the IP-address 192.168.1.20.

Source Reg	Contents	Implementation	Destination Reg	Contents
100	77	-->	1000	77
99	3200	-->	999	3200
98	820	-->	998	820
97	86280	-->	997	86280
96	14895	-->	996	14895

Example 4:

In the following example source and destination registers are addressed indirectly.

```
REGISTER_LOAD (90, 250)
REGISTER_LOAD (91, 450)
N_COPY_TO (192.168.1.20, @90, @91, 5, 3, 5)
```

The content of local registers 250 through 254 is copied to registers 450 through 454 of the controller with the IP-address 192.168.1.20.

Source Reg	Contents	Implementation	Destination Reg	Contents
250	23	-->	450	23
251	5333	-->	451	5333
252	665	-->	452	665
253	83535	-->	453	83535
254	16000	-->	454	16000

21.2.2 N_COPY_FROM

The following instruction serves to copy block-by-block the content of registers located in a remote controller to local registers.

```
N_COPY_FROM (<IP Address>, <Source Reg>, <Destination Reg>, <Number
of Registers>, <Port No>, <Mode>)
```

- **<IP-address>: IP Address** stands for the IP address of the controller to be addressed via the Ethernet network.
- **<Source Reg>:** Here, the number of the first register is specified from which the value is to be read into a local register via network the address of which is specified by **<Port No>**. This register is located on the controller which is addressed by this network instruction.
- **<Destination Reg>:** Here, the number of the local register is specified into which the content of the register specified by **<Source Reg>** is to be transmitted.
- **<Number of Registers>:** This parameter specifies the number of registers to be copied block-by-block.
- **<Port No>:** At the moment, only port number 3 is available, that is, the Ethernet network card (JetIP).
- **<Mode>:** The default mode is 1. A detailed description will be given below.

The register numbers for source and destination registers can be specified directly, indirectly or double indirectly.
The value range of the parameters for the N_COPY_FROM instruction is between 0 and 4,294,967,296.

Mode

The following modes are possible:

Mode	Meaning
The source register is directly addressed	
0	With **Number of registers > 1** the **Source Reg** remains constant (see example 1).
1	With **Number of registers > 1** the **Source Reg** will be incremented by one after each copying process (see example 2).
2	With **Number of registers > 1** the **Source Reg** will be decremented by one after each copying process (see example 3).

Mode	Meaning
The source register is indirectly addressed	
4	With **`Number of registers > 1`** the **`Source Reg`** remains constant.
5	With **`Number of registers > 1`** the **`Source Reg`** will be incremented by one after each copying process (see example 4).
6	With **`Number of registers > 1`** the **`Source Reg`** will be decremented by one after each copying process.

Example 1:

```
N_COPY_FROM (192.168.1.20, 500, 100, 3, 3, 0)
```

The content of register 500 of the controller with the IP-address 192.168.1.20 is copied to the local registers 100 through 102.

Source Reg	Contents	Implementation	Destination Reg	Contents
500	77	-->	100	77
500	77	-->	101	77
500	77	-->	102	77

Example 2:

```
N_COPY_FROM (192.168.1.20, 1000, 100, 5, 3, 1)
```

The content of registers 1000 through 1004 is copied to registers 100 through 104 of the controller with the IP-address 192.168.10.20.

Source Reg	Contents	Implementation	Destination Reg	Contents
1000	77	-->	100	77
1001	3198	-->	101	3198
1002	791	-->	102	791
1003	86320	-->	103	86320
1004	13629	-->	104	13629

Example 3:

```
N_COPY_FROM (192.168.1.20, 1000, 100, 5, 3, 2)
```

The content of registers 996 through 1000 is copied to registers 96 through 100 of the controller with the IP-address 192.168.10.20.

Source Reg	Contents	Implementation	Destination Reg	Contents
1000	77	-->	100	77
999	3200	-->	99	3200
998	820	-->	98	820
997	86280	-->	97	86280
996	14895	-->	96	14895

Example 4:

In the following example source and destination registers are addressed indirectly.

```
REGISTER_LOAD (90, 250)
REGISTER_LOAD (91, 450)
N_COPY_TO (192.168.1.20, @90, @91, 5, 3, 5)
```

The content of registers 250 through 254 is copied to registers 450 through 454 of the controller with the IP-address 192.168.10.20.

Source Reg	Contents	Implementation	Destination Reg	Contents
250	23	-->	450	23
251	5333	-->	451	5333
252	665	-->	452	665
253	83535	-->	453	83535
254	16000	-->	454	16000

22 Introduction of JetControl Web Functions

Note!

The functions which are described in this chapter are supported by JetControl 647 controllers with operating system release 3.50 and higher.

The JetControl system provides following Web functions:

- **FTP Server**: Access to directories and files of the file system on the built-in flash disk.
- **HTTP Server**: Homepages can be downloaded into the controller via FTP and accessed with any standard internet browser. Access to variables of the controller runtime system.
- **SMTP Client**: E-Mail can be sent by the controller. E-mail header parameters and message contents with variables of the controller runtime system.

All these services are based on the JetControl file system.

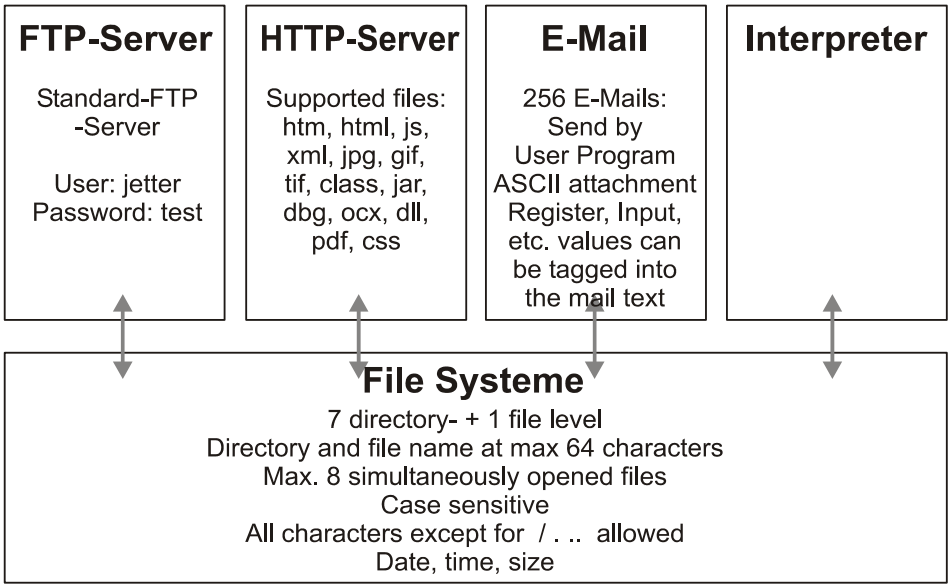

FTP-Server
Standard-FTP -Server
User: jetter
Password: test
HTTP-Server
Supported files: htm, html, js, xml, jpg, gif, tif, class, jar, dbg, ocx, dll, pdf, css
E-Mail
256 E-Mails: Send by User Program ASCII attachment Register, Input, etc. values can be tagged into the mail text
Interpreter
File Systeme
7 directory- + 1 file level
Directory and file name at max 64 characters
Max. 8 simultaneously opened files
Case sensitive
All characters except for / . .. allowed
Date, time, size

Fig. 50: File system and server, services

22.1 Initialization of Web Functions

After power-on of the controller the Web functions are initialized. Bit 0 to 5 of register 63827 report of the state of the Web function initialization respectively their availability.

Register 63827 - Bit No	Availability
0	0 = FTP server not available 1 = FTP server available
1	0 = HTTP server not available 1 = HTTP server available
2	0 = E-Mail client not available 1 = E-Mail client available
3	0 = Data file function not available 1 = Data file function available
4	0 = No Modbus/TCP 1 = Modbus/TCP has been licensed
5	0 = Modbus/TCP server not available 1 = Modbus/TCP server has been started

22.2 The FTP Server

The FTP server allows access to directories and files of the flash disk integrated into the JC-647 using an FTP client. Apart from the command line FTP cline, which comes with many PC operating systems, graphic FTP tools can be used, as well.

22.2.1 Log-In

To have access to the file system via FTP the FTP client must log in and enter the user name and password. In its original configuration the controller is provided with a user account with administrators rights:

User:	admin
Password:	admin

Via user administration of the file system, this password can be modified and new users can be added (cf. chapter 23.2 "User Administration" on page 189).

22.2.2 Key Instructions

Overview of the most important instructions supported by the FTP server:

- Transferring files from and to the controller using the instructions RETR and STOR.
- Deleting files using the instruction DELE.
- Creating and deleting directories using the instructions MKD and RMD.
- Changing the current directory using the instruction CWD.
  Example: CWD EMAIL or CWD / (back to the root directory).
- Displaying files and subdirectories of the current directory using the instruction LIST.
- Directory names are separated by a slash /, not by a backslash \.
- Case sensitivity: EMAIL `is not` Email !

22.3 HTTP Server

The HTTP server can be accessed via standard browser. Here, it may also be necessary (depending on the file system configuration) to enter the user name and password to have access to certain files.

The default file names are **`index.htm`** and **`index.html`**.

22.3.1 Supported File Types

The following file types are supported:

- *.htm, *.html
- *.js
- *.xml
- *.jpg
- *.gif
- *.tif
- *.class, *.jar
- *.dbg
- *.ocx
- *.dll
- *.pdf
- *.css

22.3.2 Serverside Includes

This function allows to display current values of controller variables (inputs, outputs, flags, registers) in an HTML page.

See also chapter 26.2 "Serverside Includes" on page 215.

22.4 The E-Mail Client

Up to 256 E-Mails can be sent from the controller via E-Mail client. The e-mail templates have to be stored in the directory **/EMAIL**.

Note!

The path **/EMAIL** has to be written in capital letters since the file system is case sensitive.

JetSym	`SPECIALFUNCTION (110, x, y)`
JetSym ST	`SYSTEMFUNCTION (110, x, y);`

x specifies the number of the register that contains the number of the E-Mail that is to be sent. The number of the E-Mail is part of the E-Mail file name: **email_0.cfg** to **email_255.cfg**.

y stands for the register number that contains the function result, e.g. error codes..

Each E-Mail consists of different sections:

[FROM]
[AN]
[CC]
[SUBJECT]
[ATTACHMENT]
[MESSAGE]

Note!

The complete path to the attached file is specified in the [ATTACHMENT] section. The attached file must be an ASCII file.

The E-Mail file itself is an ASCII file that can be created with any ASCII editor, for example the Windows Editor.

Actual controller values can be integrated into the E-Mail text via tag functions. Real time states of the process can be transmitted via e-mail using these tag functions.

For more information on e-mail refer to chapter 25 "E-Mail" on page 201.

23 File System

23.1 Properties

The file system of the JetControl 647 is for managing the built-in flash disk. This file system consists of system directories which cannot be deleted by the user and an area which is available to the user. The size of this area is:

JetControl 647	2 MBytes

Properties:
- 7 directory levels + 1 file level
- Directory or file names with a maximum length of 63 characters
- Maximum number of simultaneously opened files: 8
- Case sensitivity
- Directory names are separated by a slash / not by a backslash \.
- For directory and file names all characters are allowed, excluding "/" and "..".
- Date, time, file size (apart from system files)
- User/access administration for a maximum number of 31 locks and 32 users.

23.2 User Administration

The JetControl file system offers the possibility to define authorization for access (locks) to flash disk directories, as well as to set up users with specific permissions (keys).
Such settings can be made in various configuration files located in the directory "/System". Since these files are stored outside the normal flash disk, they will not be deleted when formatting the flash disk.
Only users with administrator rights are authorized to read and modify these files.

Note!

Administrator rights are assigned by the key "1". This assignment is permanently defined in the file system and cannot be influenced by the user.

Important!

Basically, the following rule applies: The contents of a user administration file can be read immediately after the file has been transferred. However, it will be applicable only after the next reboot of the control system.

Note!

Directories and files, for which the user does not have the required key, will not be displayed by the FTP command LIST (for displaying directory contents).

23.2.1 Flash Disk Lock File

In this file, locks can be assigned to subdirectories located on the flash disk. Only users with the corresponding key are allowed to read or write (delete) files and subdirectories located in these directories.

File

The **`flashdisklock.ini`** is a configuration file and has only one section, namely "[LOCKS]". Each subdirectory on the flash disk is specified with its lock number in an individual line. Lock numbers from 0 to 31 can be assigned. "0" means that the given subdirectory is not locked, i.e. it can be accessed without particular authorization. It is also possible to assign a name instead of a number. The name has first to be defined in the file **`keys.ini`**.

Example

```
[LOCKS]
test1=0
test1/sub1=0
test1/sub2=5
test2=user1
```

23.2.2 Key Names File

The file system supports up to 31 different user locks/keys which are consecutively numbered from 1 to 31. To provide ease of handling, a name can be assigned to each lock/key combination in this file. The names have to be unambiguously and may comprise of a maximum of 15 alphanumeric characters.
To access a subdirectory protected by a lock, users have to use the matching key. Lock and key have the same name.

File

The lock and key names are entered into the section "[KEYS]" of the file **keys.ini**. The names defined here can be used for creating locks and users once the file has been transferred and the controller has been rebooted.

Example

```
[KEYS]
KEY01=admin
KEY02=os
KEY03=
KEY04=
KEY05=
KEY06=
KEY07=
KEY08=
KEY09=
KEY10=user1
KEY11=user2
KEY12=
KEY13=
KEY14=user5
KEY15=
KEY16=
KEY17=
KEY18=
KEY19=
KEY20=
KEY21=
KEY22=
KEY23=
KEY24=
KEY25=
KEY26=
KEY27=
KEY28=
KEY29=
KEY30=
KEY31=
```

23.2.3 User File

The file system supports up to 32 users. Each user is provided with:
- an unambiguous name with up to 31 alphanumeric characters
- a password with up to 31 alphanumeric characters
- a set of up to 31 read access keys
- a set of up to write access 31 keys

The FTP and HTTP servers use the user database to control file system access.

File

The file **users.ini** is a configuration file comprising of up to 32 sections with user names ranging from "[USER1]" to "[USER32]". The user name (NAME) is mandatory, password (PW), read access keys (READKEYS) and write access keys (WRITEKEYS) are optional. The entry "SYSKEYS" is intended for future expansions and has no function at the moment.
The read and write keys are displayed as a comma separated list. They have to be entered into the list in the same format. Keys, to which names have been assigned (chapter 23.2.2 "Key Names File" on page 191), are displayed with their names. When creating the file, both names and numbers of the keys can be used.
Any user with administrator rights is allowed to modify the user file.

Note!

[USER1] must always have the name „admin“.
It is not possible to delete the user "admin" ([USER1]) nor to modify his write or read key. Only his password can be set.

Example

```
[USER1]
NAME=admin
PW=admin
READKEYS=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23
,24,25,26,27,28,29,30,31
WRITEKEYS=1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,2
3,24,25,26,27,28,29,30,31

[USER2]
NAME=OS_update
PW=jetterOS
READKEYS=1
WRITEKEYS=1

[USER3]
NAME=jetter
PW=test
```

23.3 System directories / files

On the flash disk of the JetControl 647 directories and files are located which are required for system configuration or from which system information can be read. As these directories and files are located in a memory area which is not available to the user, they cannot be deleted, but only overwritten. Even formatting the flash disk has no impact on these files.
Without administrator rights, only access to the file "/System/flashdiskinfo.txt" is granted. For access to all other directories and files administrator rights are required.

Note!

The system files have a constant date and the file size is always "0".

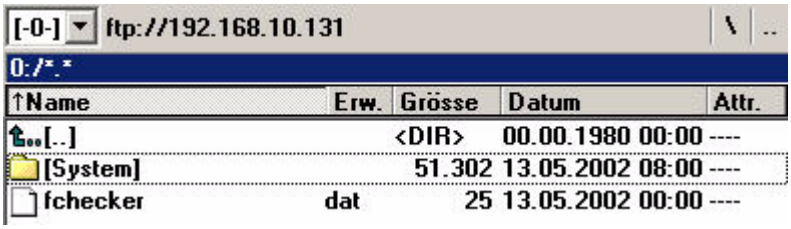


Fig. 51: Root directory of the flash disk

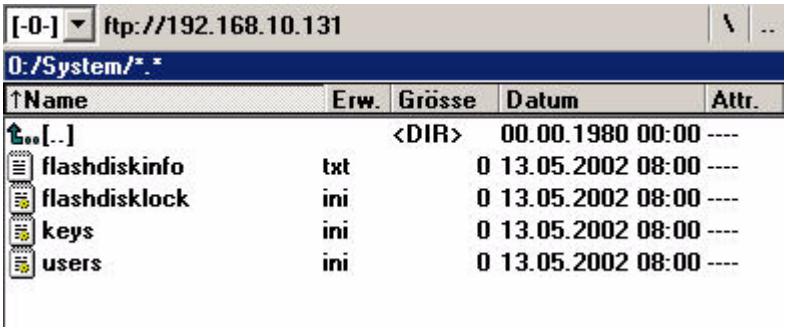


Fig. 52: Files in the directory "System"

23.3.1 The files “/System/flashdisklock.ini”, “/System/keys.ini”, and “/System/users.ini”

For more information on these files refer to chapter 23.2 "User Administration" on page 189.

23.3.2 The file "/System/flashdiskinfo.txt"

This file (the only file which can be accessed without administrator rights) provides information on the flash disk's user area.

Example:

```
Name  : flash disk
Date  : 13.05.2002
Time  : 13:07
Tracks: 33

Track   0:  sectors: 128  (used:   2 / blocked:  1 / free:  125)
Track   1:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   2:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   3:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   4:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   5:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   6:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   7:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   8:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   9:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   10:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   11:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   12:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   13:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   14:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   15:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   16:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   17:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   18:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   19:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   20:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   21:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   22:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   23:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   24:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   25:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   26:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   27:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   28:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   29:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   30:  sectors: 128  (used:   0 / blocked:  0 / free:  128)
Track   31:  sectors:  64  (used:   0 / blocked:  0 / free:   64)
Track   32:  sectors:  16  (used:   0 / blocked:  0 / free:   16)

Total:  sectors: 4048  (used: 2 / blocked: 1 / free: 4045)

Used   :     1016 byte
Blocked:      508 byte
Free   :  2054860 byte
Total:  2056384 byte
```

In this example, the file "/System/flashdiskinfo.txt" of a JetControl 647 (2 MBytes) is shown.
“`Date`" and "`Time`" indicate the time when the flash disk has been formatted.

Sector States	
used	The sector is occupied with files
blocked	The sector is no longer occupied, but can not yet be used due to administrative reasons
free	The sector is not occupied and can be used

At the end of the file statistical data are displayed.

23.4 Formatting the Flash Disk

Sometimes it might be necessary to reformat the flash disk. This can be the case if an OS release has been transferred which has a different flash disk format (e.g. when updating from OS release 2.xx to 3.xx). Or when information for flash disk administration has been destroyed (e.g. due to a voltage drop during a file operation).

23.4.1 Formatting by means of Registers

When the controller is powered-up the content of register **63794** is checked during the initialization phase. If, at that moment, this register contains the value **107**, the flash disk will be formatted and value "0" will be assigned to the register. Thus, to format the flash disk using this register proceed as follows:

1. Switch the controller on
2. Enter the value 107 into register 63794 (using JetSym, LCD HMI, ...).
3. Switch the controller off
4. Switch the controller on

→ The flash disk formatting process starts

24 FTP Server

Note!

The functions which are described in this chapter are supported by JetControl 647 controllers with operating system release 3.50 and higher.

The FTP server allows access to directories and files of the flash disk integrated into the JC-647 using an FTP client program. Apart from the command line FTP cline, which comes with many PC operating systems, graphic FTP tools can be used, as well.

The FTP server on the JetControl 647 is able to manage 2 FTP connections simultaneously. That is, up to 2 FTP client programs can be connected with the JetControl 647 at the same time. Any additional client, which tries to connect with the FTP server, will get no response to its request for establishing a connection.

24.1 Log-In

To have access to the file system via FTP, the FTP client must log in and provide its user name and password when starting the communication. In its original configuration the controller is provided with a user account with administrators rights:

User:	admin
Password:	admin

Via user administration of the file system, this password can be modified and new users can be added (cf. chapter 23.2 "User Administration" on page 189).

24.2 Supported Commands

Command	Meaning
USER	Sends the user name; is used at the beginning of the log-in process
PASS	Sends the password; is sent after USER to complete the log-in process
QUIT	Terminates the connection
PORT	Specifies the IP address and port number to which the FTP server is to connect for the next file transfer.
TYPE	Sets the transfer type; the following types are possible: • Type A with interpretation N • Typ I • Type L with 8 bits per character
MODE	Sets the transfer mode; here, only "S" (stream) is possible
STRU	Sets the file structure when transferring data; here, only "F" (file) is possible
NLST	Returns a list containing the file names of a directory
LIST	Returns a list containing the file names and file information of a directory
PWD	Returns the name of the current directory
CWD	Switches to another directory
CDUP	Moves up by one directory level
MKD	Creates a new directory
RMD	Removes a directory
STOR	Stores a file
RETR	Reads a file
DELE	Deletes a file
RNFR	Indicates the file name to be changed; must be followed by "RNTO"
RNTO	Indicates the new name of the file which has been specified by "RNFR" before.
PASV	The FTP server changes into "passive mode"

24.3 Example: Windows FTP Client

Below a description of a session using the Windows NT 4.0 FTP Client:

- Invoking the client program and opening the connection
- Log in as user "admin" with password
- Displaying the content of the current directory using "dir"
- Transferring the file "jetter1.jpg" to the JetControl using the command "put"
- Re-displaying the content of the current directory using "dir"
- Terminating the session and the FTP client program using "bye"

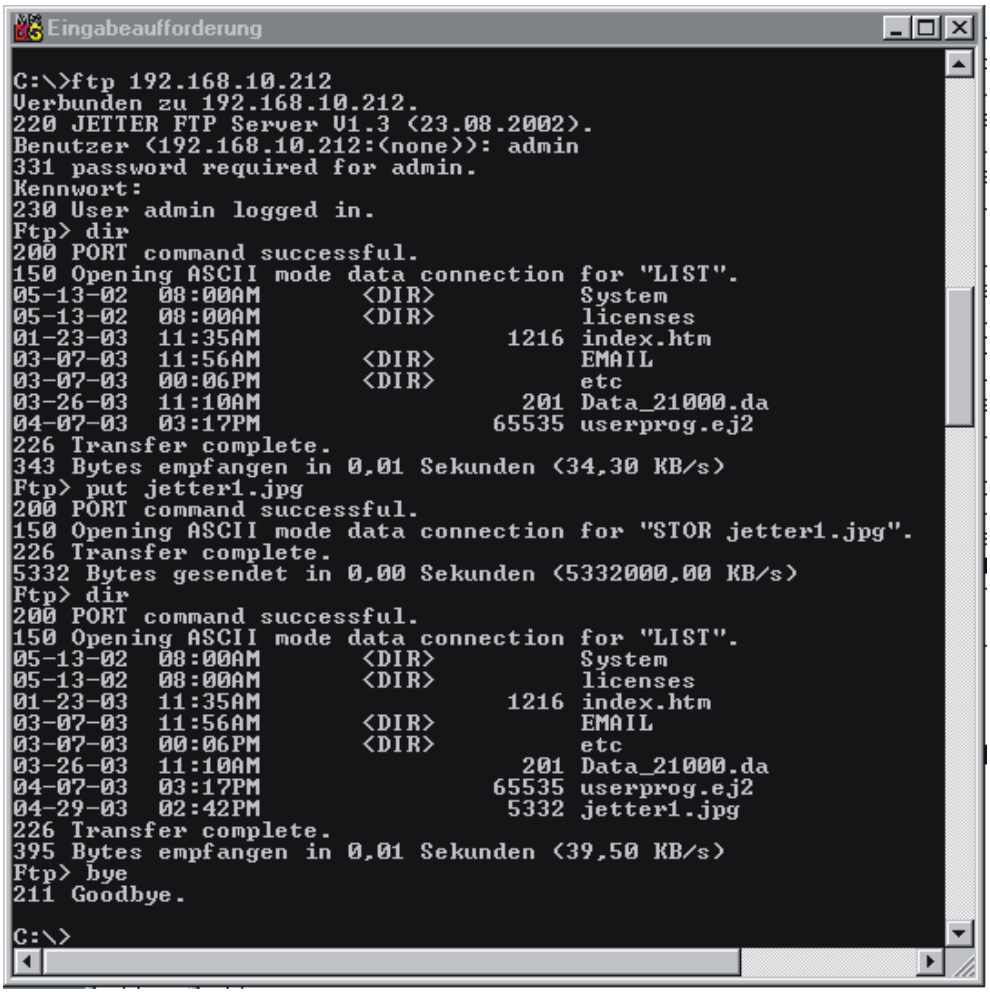


Fig. 53: Session using the Windows NT 4.0 FTP Client

25 E-Mail

Note!

The functions which are described in this chapter are supported by JetControl 647 controllers with operating system release 3.50 and higher.

E-mails are defined by means of templates which are stored to the controller in the directory **/EMAIL**. The **/EMAIL** directory has to be written in capital letters since the controller file system is case sensitive. Up to 256 different E-Mails can be created and selected for sending. e-mail transmission is triggered by **SPECIALFUNCTION / SYSTEMFUNCTION 110**.

25.1 The E-Mail INI file

The e-mail function is initialized during boot-up of the controller based on the contents of the file "/EMAIL/email.ini". This file is a pure text file which is compatible with Windows INI files.

The **email.ini** file is divided into different sections:

```
[SMTP]
IP=192.169.40.1
PORT=25
HELO=JetControl1
```

In section [SMTP] the IP address and port number of the SMTP server are specified and the parameter of the HELO command is defined.
The entries for PORT and HELO are optional. The standard port number 25 is used if no port is specified in the [SMTP] section. The HELO name has a maximum length of 63 characters. If HELO has not been defined, the entry FROM of the corresponding e-mail is used for the HELO command.
The IP address can be specified by entering a name which is, then, resolved by means of the file "/etc/hosts".

```
[DEFAULT]
MAILCFG=email_default.cfg
```

In the section [DEFAULT] the name of an e-mail template file is specified which contains default settings for e-mails. The settings made here will be used when sending an e-mail if the corresponding section in an e-mail template has not been defined.

```
[POP3]
IP=192.169.40.1
Port=110
USER=testuser
PASSWORD=test
```

Some e-mail servers require the e-mail client to log on via POP3 service before e-mails can be sent to the server.
The port number can be specified as an option. If no port number has been specified, port number 110 is used.
The IP address can be specified by entering a name which is, then, resolved by means of the file "/etc/hosts".

Note!

- Bit 2=1 of register 63827 indicates proper initialization of the e-mail client.

Following section entries are obligatory. The other definitions are optional. If no entries are made, the controller automatically uses the default values (e.g. PORT=25). In this case no entries are needed. The following sections are mandatory:

```
[SMTP]
IP=192.169.40.1
```

If the server requires a POP3 service, the following section is obligatory:

```
[POP3]
IP=192.169.40.1
USER=testuser
PASSWORT=test
```

25.2 Structure of the E-Mail Template File

The e-mail template is an ASCII file that can be created with any ASCII editor, for example the Windows Editor. The e-mail file consists of different sections.

These sections are:

[FROM]
E-mail sender

[TO]
E-mail addressee

[CC]
Additional e-mail addressee(s)

[SUBJECT]
E-mail reference

[ATTACHMENT]
Complete path and file name

[MESSAGE]
E-mail text

Remarks:

- Several addressees are separated by a semicolon “;”.
- The complete path to the attached file is specified in the [ATTACHMENT] section. The attached file must be an ASCII file. The file extension is of no significance.
- Particulars can be tagged with real time controller values that are integrated into the E-Mail text (refer to chapter 25.6: "Transmission of Controller Values", page 206).

Note

The content of the [FROM] section depends on the e-mail server. Some servers accept no domain name, other servers append the domain name automatically. For some servers the domain name must be specified. Other servers are able to handle both possibilities.

Please check with your IT administrator which function is supported by your server.

If your server does not accept a domain name, omit the domain name. Then, the server appends the domain name automatically. Only insert the **name** into the [FROM] section.

name@domainname.com

25.3 Name of the E-Mail Template File

The e-mail template is an ASCII file. It is stored in the **/EMAIL** directory. The pattern of the E-Mail file name is:

email_xyz.cfg

xyz represents the number of the e-mail template. This number can be a value between 0 and 255. Thus, a maximum of 256 e-mail templates is possible.

Examples:

email_1.cfg

etc.

email_255.cfg

25.4 Sending E-Mails

An e-mail is sent using the following command:

JetSym	`SPECIALFUNCTION (110, x, y)`
JetSym ST	`SYSTEMFUNCTION (110, x, y);`

x specifies the number of the register that contains the number of the E-mail to be sent. The number of the e-mail is part of the e-mail file name: `email_0.cfg` to `email_255.cfg`.

y stands for the register number that contains the function result. The following results are possible:

Error Code	Meaning	Possible reason
0	No error	
-1	Insufficient memory	Operating system error
-2	[FROM] not defined.	The file `email_<No>.cfg` is corrupt or cannot be found.
-3	[TO] not defined.	The file `email_<No>.cfg` is corrupt.
-4	No connection to the e-mail server or error during data transfer to the e-mail server.	No e-mail server available or wrong configuration of the `email.ini` file. Data transfer error.
-10	E-mail function not available. Bit 2 of register 63827 is not set.	Initialization error during operating system start. For example `email.ini` corrupt or cannot be found.
-12	Internal error	Operating system error

Sending an e-mail may take considerable time. Therefore, other application tasks are carried out during sending of an e-mail. However, only one function call can be carried out at a time. Tasks, which invoke this function while another task is sending an e-mail, are therefore blocked until this operation is completed.
It has to be polled whether the e-mail has been sent. For this purpose, bit 13 of register 63827 is polled.

Example:

```
SPECIALFUNCTION (110, x, y)
WHEN
   BIT_CLEAR (63827, 13)
   THEN
```

25.5 The E-Mail Default Template File

The E-Mail default template file contains sections that are used if this section is not specified in an individual E-Mail. The e-mail default template file is placed in the **/EMAIL** directory. The name of the e-mail default template file is defined in the [DEFAULT] section of the e-mail initialization file **email.ini**. The default setting of the e-mail default template file name is **email_default.cfg**.

Following default sections can be preset:

[FROM]
E-mail sender

[TO]
E-mail addressee

[CC]
Additional e-mail addressee(s)

[SUBJECT]
E-mail reference

[ATTACHMENT]
Complete path and file name

[MESSAGE]
E-mail text

25.6 Transmission of Controller Values

Actual controller values can be integrated into the parameters of the sections via tag functions. Thus, the contents respectively states of registers and text variables, inputs, outputs and flags can be transmitted via e-mail.

All tags start with the following string:

```
<JW:DTAG
```

and are closed by the characters:

```
/>
```

Between these tags variables can be defined.

25.6.1 Format Definition

The appearance of the different variables, or the number of places, can be influenced by means of the variable **Format**.

- When specifiying the format, the number of digits (characters) representing a variable can be defined by the character "#".
- Prefixing a zero "0" suppresses the output of leading zeroes (applies to the register types INT, INTX and REAL).
- Prefixing a plus sign "+" suppresses the output of a sign (applies to the register types INT, and REAL). A positive value is preceded by a “+“. A negative value is preceded by a “-“, as usual.
- Prefixing a space character suppresses the output of a space character for positive values (applies to the register types INT and REAL). A positive value is preceded by a space character. A negative value is preceded by a “-“, as usual.

Furthermore, values can be modified by a **factor** and an **offset**. If these variables have been defined, the value is multiplied by the **factor** first, then, the **offset** is added.

25.6.2 Register - Text Variable

The variable name begins with a capital **R** followed by the register number.

Example 1:

```
<JW:DTAG name="R100" type="INT" />
```

Result: The content of register 100 is inserted into the e-mail as integer number.

Example 2:

```
<JW:DTAG name="R62210" type="REAL" format="+0####.###" factor="1.5"
offset="500" />
```

Result:
The content of register 62210 is multiplied by 1.5. Then, 500 is added to the product and the result is inserted into the page with sign and at least five integer positions (if necessary, with leading zeroes) and three decimal positions.

The following formats are possible:

INT is the default type if no type has been defined.

Format	Meaning
INT	Integer decimal
INTX	Integer hexadecimal
INTB	Integer binary
BOOL	Register content = 0 -> Result: 0 Register content != 0 -> Result: 1
REAL	Floating point decimal
STRING	Text variable

25.6.3 Register - Bit Variable

The variable name begins with a capital **B** followed by the register number. The number of the bit is indicated by a preceding period.

Example 1:

```
<JW:DTAG name= "B100.2" type= "STRING" />
```

Result:
The state of bit 2 of register 100 is inserted into the e-mail as TRUE or FALSE.

Example 2:

```
<JW:DTAG name= "B100.0" />
```

Result:
The state of bit 0 of register 100 is inserted into the e-mail as 0 or 1.

The following formats are possible:

`BOOL` is the default type if no type was specified.

Format	Meaning
BOOL	Bit reset -> Display: 0 Bit set -> Display: 1
STRING	Bit reset -> Display: FALSE Bit set -> Display: TRUE

25.6.4 Flags

The variable name begins with a capital **F** followed by the flag number.

Example 1:

```
<JW:DTAG name= "F100" type= "STRING" />
```

Result:
The state of flag 100 is inserted into the e-mail as TRUE or FALSE string.

Example 2:

```
<JW:DTAG name="F100" type="STRING" format="#" />
```

Result:
The state of flag 100 is inserted into the e-mail as T or F string.

The following formats are possible:

`BOOL` is the default type if no type was specified.

Format	Meaning
BOOL	Flag reset -> Display: 0 Flag set -> Display 1
STRING	Flag reset -> Display: FALSE Flag set -> Display: TRUE

25.6.5 Inputs

The variable name begins with a capital **I** followed by the input number.

Example:

```
<JW:DTAG name="I105" type="STRING" />
```

Result:
The state of input 5 of the CPU is inserted into the e-mail as ON or OFF string.

The following formats are possible:

`BOOL` is the default type if no type was specified.

Format	Meaning
BOOL	Input inactive -> Display: 0 Input active -> Display 1
STRING	Input inactive -> Display: OFF Input active -> Display: ON

25.6.6 Outputs

The variable name begins with a capital **O** followed by the output number.

Example 1:

```
<JW:DTAG name="O105" type="BOOL" />
```

Result:
The state of output 5 of the CPU is inserted into the e-mail as 1 or 0.

Example 2:

```
<JC:DTAG name="O105" type="STRING" format="##" />
```

Result:
The state of output 5 of the CPU is inserted into the e-mail as ON or OFF.

The following formats are possible:

`BOOL` is the default type if no type was specified.

Format	Meaning
BOOL	Output inactive -> Display: 0 Output active -> Display 1
STRING	Output inactive -> Display: OFF Output active -> Display: ON

25.6.7 Indirect Access

Indirect access is realized by inserting the capital letter **P** in front of the variable name. P stands for Pointer. Access is now realized indirectly by the content of the register that is specified in the variable name.

Examples:

```
<JW:DTAG name="PR100" type="INT" />
```

Result:
The content of the register is transferred whose number was specified in register 100.

```
<JW:DTAG name="PF100" type="BOOL" />
```

Result:
The state of the flag is transferred whose number was specified in register 100.

```
<JW:DTAG name="PI100" type="BOOL" />
```

Result:
The state of the input is transferred whose number was specified in register 100.

```
<JW:DTAG name="PO100" type="BOOL" />
```

Result:
The state of the output is transferred whose number was specified in register 100.

Additionally, an offset can be defined to the value of the pointer register. This offset can be defined directly in the tag, or can also be defined indirectly by specifying a register.

Examples:

```
<JW:DTAG name="PR100 + 75" type="INT" />
```

Result:
The content of the register is transferred whose number results from the addition of the content of register 100 and the value 75.

```
<JW:DTAG name="PO100 + R75" type="BOOL" />
```

Result:
The state of the output is transferred whose number results from the addition of the content of the two registers 100 and 75.

25.7 Peak Values

For the individual e-mail sections the following number of characters is allowed:

[FROM]	63
[TO]	127
[CC]	127
[SUBJECT]	127
[MESSAGE]	8191

25.8 E-Mail Tracking Register

With the help of two registers the user can track whether an e-mail has been sent.

A description of these registers is given in chapter 32.7: "E-mail Tunction", page 304.

26 HTTP Server

Note!

The functions which are described in this chapter are supported by JetControl 647 controllers with operating system release 3.50 and higher.

The HTTP server can be accessed via standard browser. The browser is for reading and displaying files which have been downloaded to the controller via FTP. To do so, it may also be necessary (depending on the file system configuration) to enter the user name and password to have access to certain files.

The default file names are **`index.htm`** and **`index.html`**.

26.1 Supported File Types

The following file types are supported:

- *.htm, *.html, *.shtml
- *.txt, *.ini
- *.gif, *.tif, *.tiff, *.wbmp
- *.jpg, *.jpe, *.jpeg, *.png
- *.xml
- *.js, *.jar, *.java, *.class, *.cab
- *.ocx
- *.pdf, *.zip, *.doc, *.rtf
- *.css
- *.wml, *.wmlc, *.wmls, *.wmlsc

26.2 Serverside Includes

- Current controller values can be displayed in an HTML page using the "Serverside Includes" (SSI) function in the HTTP server.
- To do so, a **`"Name Space Tag"`** has to be defined at the beginning of the HTML page that is to contain the controller values. This name space tag defines the **`"name space"`** used in the page.
- When the page is loaded into the browser, the HTTP server replaces the **`"Data Tags"`** in the "Body" section of the page once by current controller values.

Note!

To refresh the controller values, the HTML page must be reloaded.

26.2.1 Name Space Tag

The **"Name Space Tag"** must be the first entry in the HTML file. Its structure is as follows:

```
<NS:DTAG xmlns:NS=http://jetter.de/ssi/jetcontrol/>
```

Here, NS represents the **"Name Space"**. A character string with a maximum length of 63 characters can be chosen for the name space. The **"Name Space"** introduced here will be used again for the subsequent **"Data Tags"**.

The remaining parts of the line are fixed and have to be specified in exactly the same manner.

In the following examples, JC is used for **"Name Space"**.

26.2.2 Data Tag

All data tags begin with the following string:

```
<NS:DTAG
```

They are completed by:

```
/>
```

Variables can be defined within these delimiters.

Format Definition

The appearance of the different variables, or the number of places, can be influenced by means of the variable **Format**.

- When specifiying the format, the number of digits / characters representing a variable can be defined by the character "#".
- Prefixing a zero "0" suppresses the output of leading zeroes (applies to the register types INT, INTX and REAL).
- Prefixing a plus sign "+" suppresses the output of a sign (applies to the register types INT, and REAL). A positive value is preceded by a “+“. A negative value is preceded by a “-“, as usual.
- Prefixing a space character suppresses the output of a space character for positive values (applies to the register types INT and REAL). A positive value is preceded by a space character. A negative value is preceded by a “-“, as usual.

Furthermore, values can be modified by a **factor** and an **offset**. If these variables have been defined, the value is multiplied by the factor first, then, the offset is added.

Register - Text Variable

The variable name begins with a capital **R** followed by the register number.

Example 1:

```
<JC:DTAG name= "R100" type= "INT" />
```

Result:
The content of register 100 is inserted into the e-mail as integer number.

Example 2:

```
<JC:DTAG name= "R62210" type= "REAL" format="+0####.### factor="1.5"
offset="500" />
```

Result:
The content of register 62210 is multiplied by 1.5. Next, 500 is added to the product and the result is inserted into the page with sign and at least four integer positions (if necessary with leading zeroes) and three decimal positions.

The following formats are possible:

INT is the default type if no type has been defined.

Format	Meaning
INT	Integer decimal
INTX	Integer hexadecimal
INTB	Integer binary
BOOL	Register content = 0 -> Result: 0 Register content != 0 -> Result: 1
REAL	Floating point decimal
STRING	Text variable

Register - Bit Variable

The variable name begins with a capital **B** followed by the register number. The number of the bit is indicated by a preceding period.

Example 1:

```
<JC:DTAG name= "B100.2" type= "STRING" />
```

Result:
The state of bit 2 of register 100 is represented as TRUE or FALSE.

Example 2:

```
<JC:DTAG name= "B100.0" />
```

Result:
The state of bit 0 of register 100 is represented as 0 or 1.

The following formats are possible:

`BOOL` is the default type if no type has been specified.

Format	Meaning
BOOL	Bit reset -> Display: 0 Bit set -> Display: 1
STRING	Bit reset -> Display: FALSE Bit set -> Display: TRUE

Flags

The variable name begins with a capital **F** followed by the flag number.

Example 1:

```
<JC:DTAG name= "F100" type= "STRING" />
```

Result:
The state of flag 100 is inserted into the page as TRUE or FALSE string.

Example 2:

```
<JC:DTAG name="F100" type="STRING" format="#" />
```

Result:
The state of flag 100 is inserted into the page as T or F string.

The following formats are possible:

`BOOL` is the default type if no type has been specified.

Format	Meaning
BOOL	Flag reset -> Display: 0 Flag set -> Display 1
STRING	Flag reset -> Display: FALSE Flag set -> Display: TRUE

Inputs

The variable name begins with a capital **I** followed by the input number.

Example:

```
<JC:DTAG name="I105" type="STRING" />
```

Result:
The state of input 105 is inserted into the page as ON or OFF string.

The following formats are possible:

BOOL is the default type if no type has been specified.

Format	Meaning
BOOL	Input inactive -> Display: 0 Input active -> Display 1
STRING	Input inactive -> Display: OFF Input active -> Display: ON

Outputs

The variable name begins with a capital **O** followed by the output number.

Example 1:

```
<JC:DTAG name="O105" type="BOOL" />
```

Result:
The state of output 105 CPU is represented as 1 or 0.

Example 2:

```
<JC:DTAG name="O105" type="STRING" format="##" />
```

Result:
The state of output 105 is represented as ON or OFF.

The following formats are possible:

`BOOL` is the default type if no type has been specified.

Format	Meaning
BOOL	Output inactive -> Display: 0 Output active -> Display 1
STRING	Output inactive -> Display: OFF Output active -> Display: ON

Indirect Access

Indirect access is realized by inserting the capital letter **P** in front of the variable name. P stands for Pointer. Access is now realized indirectly by the content of the register that is specified in the variable name.

Examples:

```
<JC:DTAG name="PR100" type="INT" />
```

The content of the register is transferred whose number was specified in register 100.

```
<JC:DTAG name="PF100" type="BOOL" />
```

The state of the flag is transferred whose number was specified in register 100.

```
<JC:DTAG name="PI100" type="BOOL" />
```

The state of the input is transferred whose number was specified in register 100.

```
<JC:DTAG name="PO100" type="BOOL" />
```

The state of the output is transferred whose number was specified in register 100.

Additionally, an offset can be defined to the value of the pointer register. This offset can be defined directly in the tag, or can also be defined indirectly by specifying a register.

Examples:

```
<JC:DTAG name="PR100 + 75" type="INT" />
```

The content of the register is transferred whose number results from the addition of the content of register 100 and the value 75.

```
<JC:DTAG name="PO100 + R75" type="BOOL" />
```

The state of the output is transferred whose number results from the addition of the content of the two registers 100 and 75.

26.2.3 Example Page

```
<JC:DTAG xmlns:JC="http://jetter.de/ssi/jetcontrol" />
<html>

<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1252">
<meta name="GENERATOR" content="Microsoft FrontPage 4.0"
<meta name="ProgId" content="FrontPage.Editor.Document">
<title>Index</title>
</head>

<body>
Hello world,  
<p>Current controller values can be inserted into an html page like
this: </p>
<p>Register 100 = <JC:DTAG name= "R100" type="INT" format="+####" />,
or Hex: 0x<JC:DTAG name="PR100+10" type="INTX" format="0###" />,
or better like this <JC:DTAG name="R100" type="BOOL" />, if only
Boolean is queried.
But binary is also possible: <JC:DTAG name="R100" type="INTB"
format="########" />b. </p>
<p>Strings can also be defined „<JC:DTAG name="R1000" type="STRING" /
>" . </p>
<p>Real numbers look like this: <JC:DTAG name="R62210" type="REAL" />
or <JC:DTAG name="R62210" type="REAL" factor="1.3" format="###.##" /
>. </p>

<p>The value of a flag is represented as follows: <JC:DTAG name="F10"
/>
or <JC:DTAG name="PF100" type="STRING" />. </p>
<p>By analogy for inputs and outputs: <JC:DTAG name="PI130"
type="BOOL" />
or <JC:DTAG name="O105" type="STRING" />. </p>
<p>R100 = <JC:DTAG name="R100" type="INT" format="+0##########" />
  </p>
<p>Best regards </p>
<p>Your JetControl</p>
</body>

</html>
```

26.3 Visualization via Internet browser

As communication component for transferring controller data to allow browser-based visualization a Java applet and an ActiveX are available.

For more information please contact a Jetter AG sales representative.

27 RemoteScan

The configurable RemoteScan function is used to cyclically copy register/variable contents from the JetControl to registers/variables of network nodes. On the other hand, they can be read by the nodes and copied into JetControl registers/variables. The RemoteScan function can be accessed by using special/system functions 80, 81, and 82.

At the moment, RemoteScan via Modbus/TCP is supported only.

27.1 Special/System Functions

27.1.1 Configuring RemoteScan

JetSym	`SPECIALFUNCTION (80, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (80, Parameter1, Parameter2);`

Operating principle: To configure RemoteScan special/system function 80 is used. However, this function does not start communication.

Parameter 1: Number of the first register / address of the first variable of a description block.

Offset	Meaning	
0	Protocol	5 = Modbus/TCP
1	Quantity of following communication units	1 .. 10
	Communication unit 1	
2	Address	Modbus/TCP: IP-address
3	Port number	Modbus/TCP: 502
4	Update rate	10 .. 65535 ms
5	Quantity of output registers	0 .. 125
6	Output source register number	local
7	Output destination register number	remote
8	Quantity of input registers	0 .. 125
9	Input source register number	remote
10	Input destination register number	local

	11	Number of the first register of the status register block	
	12	Timeout	in ms
		Communication unit 2	
	13	Address	Modbus/TCP: IP-address
	..	..	

Parameter 2: Number of the register / address of the variable resulting from the function.

Potential errors:	Error code	Meaning
	0	No error
	-1	Protocol not supported
	-2	Set quantity of communication units > 10
	-3	Invalid address or port number
	-4	Invalid register number
	-10	RemoteScan is already running

Note!

- With modules without input or output registers the corresponding quantity has to be set to 0.
- If the configuration comprises inputs and outputs, the outputs are sent first, then the inputs are read.
- The address (always the first entry of a communication unit) has to be specified directly. It cannot be specified using names.

Status register block

The number of the first register of the status register block, consisting of 3 consecutive registers, has to be specified in the description block of each communication unit (Offset 11, 24). Error messages of this communication unit are stored to this block when RemoteScan is running.

The status register block has the following structure:

Reg. Offset	Meaning	
0	Status (bit-coded)	Bit # Meaning 0 Scan is running — is set after each update cycle 1 An error has occurred — is set each time an error has occurred
1	Error code	The code of the last error is displayed. Value Meaning 0 No error < 0 Application-specific error Modbus/TCP: Value Meaning -1 Error in the network driver -2 Error in the connection management -3 Error when sending output registers -4 Error when reading input registers -5 Exception response -6 Error when receiving the response -7 Wrong transaction ID -8 Timeout 101 Timeout 102 Error when reading/writing local registers 103/104 Error in the lower-level communication layer
2	Number of errors	The number is incremented each time an error occurs.

Note!

It is useful to initialize the contents of the status register blocks with 0 before starting the RemoteScan.

27.1.2 Starting RemoteScan

JetSym	`SPECIALFUNCTION (81, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (81, Parameter1, Parameter2);`

Operating principle	Function 81 is used to start a RemoteScan that has been configured using function 80.
Parameter 1:	No parameters are transferred to this function. Thus, the content of parameter 1 is of no significance.
Parameter 2:	This function always returns the value 0 as result.
Potential errors:	None

27.1.3 Stopping RemoteScan

JetSym	`SPECIALFUNCTION (82, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (82, Parameter1, Parameter2);`

Operating principle	Function 82 is used to stop a running RemoteScan. When doing so, all possibly existing communication connections are closed.
Parameter 1:	No parameters are transferred to this function. Thus, the content of parameter 1 is of no significance.
Parameter 2:	This function always returns the value 0 as result.
Potential errors:	None

Note!

Execution of this function may take a relatively long time, depending on the configuration, since it waits until all currently running transfers will be terminated.

27.2 Registers

A description of these registers is given in chapter 32.8 "RemoteScan Function" on page 305.

28 Modbus/TCP

28.1 Modbus/TCP Server

With a valid license and after a successful start of the Modbus/TCP server, an external client can access registers, inputs and outputs. In doing so, 4 connections may be opened at the same time.

Note!

Since only registers with a width of 16 bits can be transferred via Modbus/TCP, access to the high-order 16 bits of JetControl registers is not possible. When receiving register values, sign extension to 32 bits will not be carried out.

28.1.1 Supported Commands - Class 0

read multiple registers (fc 3)

Reading register blocks.
The number of the start register corresponds to the register number within the JC-647.

write multiple registers (fc 16)

Writing register blocks.
The number of the start register corresponds to the register number within the JC-647.

28.1.2 Supported Commands - Class 1

read coils (fc 1)

Reading outputs.
The output number has to be transferred in the internal format of the JC-647 controller.

JetSym – Application-specific numbering format	JC-647 – Internal numbering format
101 .. 164	0 .. 0x3F
201 .. 264	0x40 .. 0x7F
etc.	etc.

read input discretes (fc 2)

Reading inputs.
The input number has to be transferred in the internal format of the JC-647 controller.

JetSym – Application-specific numbering format	JC-647 – Internal numbering format
101 .. 164	0 .. 0x3F
201 .. 264	0x40 .. 0x7F
etc.	etc.

read input registers (fc 4)

Reading inputs summarized in 16-bit words.

Inputs	Register Number
101 .. 116	0
201 .. 216	2
301 .. 316	4
etc.	etc.

write coil (fc 5)

Activating/deactivating an individual output.
The output number has to be transferred in the internal format of the JC-647 controller.

JetSym – Application-specific numbering format	JC-647 – Internal numbering format
101 .. 164	0 .. 0x3F
201 .. 264	0x40 .. 0x7F
etc.	etc.

write single register (fc 6)

Entering values into the low-order 16 bits of a JC-647 register.

28.1.3 Supported Commands - Class 2

force multiple coils (fc 15)

Activating/deactivating several outputs.
The output number has to be transferred in the internal format of the JC-647 controller.

JetSym – Application-specific numbering format	JC-647 – Internal numbering format
101 .. 164	0 .. 0x3F
201 .. 264	0x40 .. 0x7F
etc.	etc.

read / write registers

Writing and simultaneously reading registers.
The number of the start register corresponds to the register number within the JC-647. First, the registers polled by the client are read, then, the registers transferred from the client are stored.

28.2 Modbus/TCP Client

The Modbus/TCP client in JetControl 647 controllers supports Class 0 Conformance (see chapter 28.1.1 "Supported Commands - Class 0" on page 229). This means that commands for reading and writing multiple registers are used. Up to 125 registers with a width of 16 bits can be transmitted in one frame. When sending 32-bit registers only the lower-order 16 bits are transmitted. When assigning incoming register values to the JetControl-internal 32-bit registers sign extension to 32 bits will not be carried out.
As protocol ID "0" is used, as unit ID "1". Assignment of sent and received frames is carried out using the transaction ID.
Connections to 11 different servers may be opened at the same time.

28.2.1 RemoteScan

This function is for cyclically transferring the inputs and outputs 14001 through 19999 that are combined in the 16-bit registers 66000 through 66999 from and to the configured servers. One connection is established to each server (IP address and port) irrespective of the number of communication units configured on this server. If several communication units are configured on one server, accesses are serialized since servers, as a rule, do not support "command pipelining". If several servers have been configured, communication is carried out in parallel.
(cf. chapter 27 "RemoteScan" on page 225)

Overlaying of inputs/outputs on registers

Registers	Inputs / Outputs
66000	14001 - 14016
66001	14017 - 14032
66002	14033 - 14048
etc.	etc.
66375	19999
66998	No assignment
66999	No assignment

Note!

Since the registers and the inputs/outputs overlaid on them merely are memory cells located in the RAM, and no direct mapping to hardware takes place, it is not determined whether a register contains inputs or outputs. Assignment is made not until configuration in the communication units takes place.

28.2.2 Special/System Functions

As acyclic transmission channel to a Modbus/TCP server the special functions 65 (reading registers) and 66 (writing registers) can be used (the functions are available independent of RemoteScan).
While one of these two special functions is being carried out simultaneous calls of this functions in other tasks are blocked until this function will be terminated.
These functions establish a connection to the specified server, transmit the desired data and clear down the connection. In case a connection exists that has been established by RemoteScan, this connection will be used. Setting-up and clearing-down the connection is not required.
The IP address is always to be specified directly. It cannot be specified using names.

Important!

It is not advisable to issue TaskBreak or TaskRestart instructions for this task or to restart the program through JetSym while one of these functions is carried out since in such a case the connection remains established which may block additional transmissions.

Special Function 65: Reading registers

JetSym `SPECIALFUNCTION (65, Parameter1, Parameter2)`

JetSym ST `SYSTEMFUNCTION (65, Parameter1, Parameter2);`

Operating principle: Using this function, a register block from a Modbus/TCP server is copied to the registers/variables of the local memory.

Parameter 1: Number of the first register / address of the first variable of a description block.

Offset	Meaning	
0	IP address	
1	Port number	502
2	Timeout	in ms
3	Number of the source register	remote
4	Number of the designation register	local
5	Quantity of Registers	1 .. 125

Parameter 2: Number of the register / address of the variable resulting from the function.

Potential errors:

Error code	Meaning
0	No error
-1 or -2	Error during connection set-up
-4	Error during data transfer
-5	Error message from server
-8	Timeout
-10	No Modbus/TCP license

Special Function 66: Writing to Registers

JetSym	`SPECIALFUNCTION (66, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (66, Parameter1, Parameter2);`

Operating principle: Using this function, the content of registers/variables of the local memory is copied to a register block a Modbus/TCP server.

Parameter 1: Number of the first register / address of the first variable of a description block.

Offset	Meaning	
0	IP address	
1	Port number	502
2	Timeout	in ms
3	Number of the source register	local
4	Number of the designation register	remote
5	Quantity of Registers	1 .. 125

Parameter 2: Number of the register / address of the variable resulting from the function.

Potential errors:

Error code	Meaning
0	No error
-1 or -2	Error during connection set-up
-3	Error during data transfer
-5	Error message from server
-8	Timeout
-10	No Modbus/TCP license

29 Data Files

The latest register values and flag states can be written into a file or read-out of a file with the help of some special functions. This read/write process is controlled by the application program.
The file format is identical to the format of the "data dump" files created by JetSym (see chapter 29.3 "File Format" on page 242).
The file names consist of two constant parts and a register content. So, files can be selected by different register values (see chapter 29.1.2 "File Names" on page 237).
Written files are stored to the root directory (highest directory level) of the flash disk. Files to be read must also be located in the root directory. Access to data files is carried out with administrator rights and cannot be restricted.

29.1 Special/System Functions

29.1.1 Implementation

Since file operations may take considerable long time, especially with large files, other application tasks are processed while one of the file operations is running. However, only one function can be processed at a time. Tasks which invoke one of these functions while a file operation of another task is running are therefore blocked until this operation is completed.
That implies that data consistency of value blocks to be written or read is not ensured. Data consistency has to be ensured by accordingly programming the application program.
The state of the currently running operation can be polled through the registers specified below.

29.1.2 File Names

File names always start with "Data_" followed by a numerical value and the extension "da". The numerical value for drawing a distinction between various files is acquired from the parameter register of the special functions. When doing so, only positive values are allowed.

Examples:

```
Data_123456789.da
Data_0.da
```

Note!

Observe capitalization. The file system is case sensitive.

29.1.3 Saving values – Creating a file

JetSym	`SPECIALFUNCTION (90, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (90, Parameter1, Parameter2);`

Operating principle: Function 90 is for creating a new data file and inserting a selectable register or flag block into this file.

Parameter 1: Number of the first register / address of the first variable of a description block.

Offset	Meaning
0	File Name
1	Variable type 1 = Register 3 = Flag
2	Number of the first register or flag
3	Number of the last register or flag

Parameter 2: Number of the register / address of the variable resulting from the function.

Potential errors:

Error code	Meaning
0	No error
-1	Error when creating file (e.g. disk full)
-2	Error when writing data
-4	Error when closing file
-6	Invalid register/flag number
-10	Data file feature not available
-20	Internal OS error

29.1.4 Saving values – Appending to a file

JetSym `SPECIALFUNCTION (91, Parameter1, Parameter2)`

JetSym ST `SYSTEMFUNCTION (91, Parameter1, Parameter2);`

Operating principle: Function 91 is for appending a selectable register or flag block to an existing file. In case this file does not exist, a new file will be created.

Parameter 1: Number of the first register / address of the first variable of a description block.

Offset	Meaning
0	File Name
1	Variable type 1 = Register 3 = Flag
2	Number of the first register or flag
3	Number of the last register or flag

Parameter 2: Number of the register / address of the variable resulting from the function.

Potential errors:

Error code	Meaning
0	No error
-1	Error when opening or creating file
-2	Error when writing data
-4	Error when closing file
-6	Invalid register/flag number
-10	Data file feature not available
-20	Internal OS error

29.1.5 Reading values from a file

JetSym	`SPECIALFUNCTION (92, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (92, Parameter1, Parameter2);`

Operating principle

Special function 92 is for reading register values and flag states out of a data file and entering them into the corresponding registers or flags. The information is processed in the order specified by the content of the file.

Parameter 1:

Number of the first register / address of the first variable of a description block.

Offset	Meaning
0	File Name

Parameter 2:

Number of the register / address of the variable resulting from the function.

Potential errors:

Error code	Meaning
0	No error
-1	Error when opening the file
-3	Error when reading data
-4	Error when closing file
-10	Data file feature not available
-20	Internal OS error

29.1.6 Deleting a file

JetSym	`SPECIALFUNCTION (96, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (96, Parameter1, Parameter2);`

Operating principle	Function 96 is for deleting a data file from the flash disk.
Parameter 1:	Number of the first register / address of the first variable of a description block.

Offset	Meaning
0	File Name

Parameter 2:	Number of the register / address of the variable resulting from the function.

Potential errors:

Error code	Meaning
0	No error
-5	Error when deleting the file
-10	Data file feature not available
-20	Internal OS error

29.2 Registers

A description of these registers is given in chapter 32.9 "Data Files" on page 306.

29.3 File Format

These files are pure text files with one line for each entry. The entries are to be terminated with "carriage return / line feed". Comment lines are allowed.
A data file is to start with the file identifier "SD1001".
Data lines start with an identifier for the variable. Now follows the number of the variable also separated by a blank or tab. Now follows the value of the variable also separated by a blank or tab.
The IDs at the beginning of a line must not be indented.

Variable ID	Variable type
FS	Flags
QS	Floating point numeral register
RS	Integer register

All lines that do not start with such a variable ID are regarded as comment lines with the exception of the first line containing the file ID.

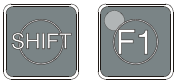
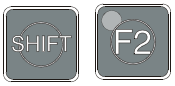
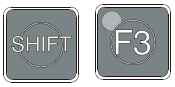
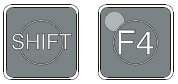
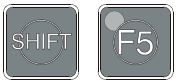
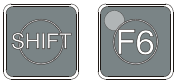
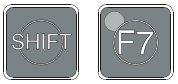
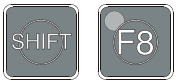
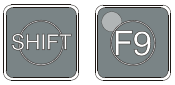
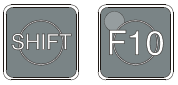
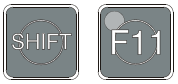
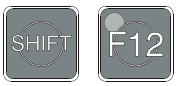
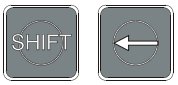
Example:

```
SD1001
; JC-647 DATA FILE - Jetter AG
FS 111 1
This is a comment
RS 20112 110
  FS 113 1
QS 65024 -3.141593
QS 65025 6.789e-05
```

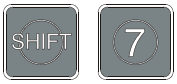
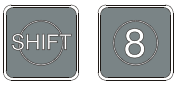
The third line from below is also a comment line since the variable's ID ("FS") does not stand at the beginning of the line.

30 Special Flags

Control of HMI LEDs			
Special Flag	**LED, Key**	**Special Flag**	**LED, Key**
2224	LED of	**2230**	LED of
2225	LED of	**2231**	LED of
2226	LED of	**2232**	LED of
2227	LED of	**2233**	LED of
2228	LED of	**2234**	LED of
2229	LED of	**2235**	LED of

Scanning HMI Keys			
Special Flag	**LED, Key**	**Special Flag**	**LED, Key**
Function Keys			
2201		**2181**	
2202		**2182**	
2203		**2183**	
2204		**2184**	
2205		**2185**	
2206		**2186**	
2207		**2187**	
2208		**2188**	
2209		**2189**	
2210		**2190**	
2211		**2191**	
2212		**2192**	
Special function keys (does not apply to LCD 27)			
2214		**2193**	
2213		**2194**	

Scanning HMI Keys			
Special Flag	**LED, Key**	**Special Flag**	**LED, Key**
2215	R	**2195**	SHIFT + R
2216	I/O	**2196**	SHIFT + I/O
2217	=	**2197**	SHIFT + =
2218	C	**2198**	SHIFT + C
2219	ENTER	**2199**	SHIFT + ENTER
2220	–	**2221**	SHIFT + –
2222	.	**2223**	SHIFT + .
2200	SHIFT		
Numerical Keys			
2160	0	**2170**	SHIFT + 0
2161	1	**2171**	SHIFT + 1
2162	2	**2172**	SHIFT + 2
2163	3	**2173**	SHIFT + 3
2164	4	**2174**	SHIFT + 4
2165	5	**2175**	SHIFT + 5
2166	6	**2176**	SHIFT + 6

Scanning HMI Keys			
Special Flag	**LED, Key**	**Special Flag**	**LED, Key**
2167		**2177**	
2168		**2178**	
2169		**2179**	
LCD 27			
2209		**2210**	
2211		**2212**	
LCD 16 - NUM 25			
2206		**2186**	
2207		**2187**	
2208		**2188**	
2209		**2189**	
2210		**2190**	

(Error) messages via special flags	
Special Flag	**Function**
2104	Timeout during last slave register access
2105	At least one timeout since reset during slave register access
2110	Timeout during last network access (JETWay)
2111	At least one timeout since reset during network access (JETWay)
2115	Network error (Ethernet). The flag is cleared at program restart or by the user.
2136	Illegal destination for GOTO or CALL indirect
2137	Current instruction would cause stack overflow; task was interrupted
2138	Current instruction would cause stack underflow; task was interrupted
2139	No application program or CRC error.
2140	OP-code error
2141	Syntax error in the interpreter
2144	Error message from outputs
2145	Real-time clock found
2146	Battery for real-time clock is ok
2147	Battery for register-RAM is almost flat

Task Control	
Special Flag	**Function**
2056	Task switch after elapsed task timeout time (reg. 61804) in milliseconds
2057	Task switch in case of GO TO instruction
2058	Task switch in case of IF instruction (not fulfilled)
2112	Interface to PC and HMI depending on task Priority assignment of communication tasks

Arithmetic operations	
Special Flag	**Function**
2048	Automatic switchover from integer to real in case of division (remains on real mode) 0 = ON
2049	Always real mode
2076	Shift Carry The last register bit shifted out by the instructions SHIFT_LEFT and SHIFT_RIGHT.

HMI Control	
Special Flag	**Function**
2051	The USER_INPUT instruction is active
2053	USER_INPUT terminated by timeout

HMI Restrictions	
Special Flag	**Function**
2052	Input via HMI keys disabled
2054	ENTER key does not switch to monitor mode
2096	Displaying registers possible 0 = disabled
2097	Displaying flags possible 0 = disabled
2098	Displaying outputs possible 0 = disabled
2099	Displaying inputs possible 0 = disabled
2100	Modifying registers possible 0 = disabled
2101	Modifying flags possible 0 = disabled
2102	Modifying outputs possible 0 = disabled
2103	Permanent display of inputs not possible

Interface Activity Monitoring		
Special Flag	**Interface**	**Function**
2080	NET 2	OS flag 0 = no JETWay activity 1 = JETWay activity
2081	NET 2	User flags 0 = no JETWay activity 1 = can be set by the user if flag 2080 = 1
2088	Ethernet	OS flag 0 = no JETIP activity 1 = JETIP activity
2089	Ethernet	User flags 0 = no JETIP activity 1 = can be set by the user if flag 2088 = 1
2090	PC	OS flag 0 = no pcom5 activity 1 = pcom5 activity
2091	PC	User flags 0 = no pcom5 activity 1 = can be set by the user if flag 2090 = 1
2092	LCD	OS flag 0 = no pcom5 activity 1 = pcom5 activity
2093	LCD	User flags 0 = no pcom5 activity 1 = can be set by the user if flag 2092 = 1
2094	NET 1	OS flag 0 = no JETWay activity 1 = JETWay activity
2095	NET 1	User flags 0 = no JETWay activity 1 = can be set by the user if flag 2094 = 1

31 Overview of Special Registers

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Operating system error messages			
61473	Modified	Operating system error	1) 0 .. 255 2) 0 3) Page 271
61477	Modified	Operating system error	1) 0 .. 255 2) 0 3) Page 272
61478	Same	Error, operating system messages	1) 0 .. 255 2) 0 3) Page 273
61530	Same	Task number of the OPC error	1) 0 .. 255 2) not defined
61672	Same	Error messages during access to slave modules	1) 0 .. 255 2) 0 3) Page 274
63770	New	Numerator with division by 0	1) 0 .. 65.535 2) 0 3) Page 274
63771	New	Program address which has caused division by 0	1) 0 .. (256 * 1024) 2) 0 3) Page 275
63784	New	Number of error messages in the fault memory since reset	1) 0 .. +8.388.607 2) 0
63785	New	Selection of error message	1) 0 .. +8.388.607 2) 0
63786	New	Contents of the error message selected by reg. 63785	1) 0 .. 65.535 2) 0

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Operating system messages			
61952	Same	Application program runtime in seconds	1) -8.388.608 .. +8.388.607 2) 0
61953	Same	Controller runtime since reset in seconds	1) -8.388.608 .. +8.388.607 2) 0
61954	Same	Controller runtime in time base units	1) -8.388.608 .. +8.388.607 2) 0
61956	Same	Controller runtime since reset in ms	1) -8.388.608 .. +8.388.607 2) 0
62977	Same	Operating system version * 100	1) 0 .. 65.535 2) 0
63787	New	Current build level	1) 0 .. 65.535 2) Current build level 3) Page 275

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Access to controller modules (SV, SM, PID) Flag 2105 = 1 reports error			
62150	Same	Access error controller module: Slot number - 1	1) 0 .. 255 2) 0
62151	Same	Access error controller module: Axis number - 1	1) 0 .. 255 2) 0
62152	Same	Access error controller module: Register Number	1) 0 .. 255 2) 0

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Task Control			
61449	Same	Prioritised task	1) 0 .. 255 2) 0
61467	Same	Task switch conditions	1) 0 .. 255 2) 0 3) Page 276
61474	Modified	Multitasking control	1) 0 .. 255 2) 0 3) Page 277
61610	Same	Highest task number in the program	1) 0 .. 255 2) 0
61773	Same	Min. cycle time in 1 ms increments	1) +/- 32.767 2) 0
61774	Same	Max. cycle time in 1 ms increments	1) +/- 32.767 2) 0
61777	Same	Cycle time of all tasks in 1 ms increments	1) +/- 32.767 2) 0
61804	Same	Task timeout time in 1 ms increments	1) 0 .. 255 2) 20
185000 .. 185099	Originally register 61531 .. 61562	Task status	1) 0 .. 255 2) Status 3) Page 277
185200 .. 185299	Originally register 61708 .. 61739	Task time register for DELAY	1) 0 .. 2.147.483.647 2) 0

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Computing speed JC-647			
63796	New	Computing speed JC-647	1) 0 .. 1 2) 0 3) Page 275

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of HMIs (LCD Display) Display and input of fixed-point numbers			
61451	Modified	Does not exist Field width for floating point numbers	
61452	New	Entering Fixed-Point Numbers Number of decimal places	1) 0 .. 255 2) 0 3) Page 279
61453	Modified	Field width for integer numbers	1) 0 .. 255 2) 0 3) Page 278
61454	Modified	Does not exist Left-justified representation	
61455	Modified	Field width for user input	1) 0 .. 255 2) 8 3) Page 282
61781	New	Sign suppression	1) 0 .. 1 2) 0 3) Page 279
61786	New	Entering Fixed-Point Numbers Divisor	1) 0 .. 65.535 2) 1 3) Page 280
61976	Same	Suggested value for user input	1) -8.388.608 .. +8.388.607 2) 0 3) Page 281
63809	New	User input of fixed-point numbers Divisor	1) 0 .. 65.535 2) 1 3) Page 282
63810	New	User input of fixed-point numbers Number of decimal places	1) 0 .. 255 2) 0 3) Page 283

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of HMIs (LCD Display) Display			
61448	Same	Display language 0 = German 1 = English	1) 0 .. 255 2) 1 3) Page 285
61461	Same	"Delete End Of Line" character	1) 0 .. 255 2) 36 3) Page 290
61462	Same	"Delete Screen" character	1) 0 .. 255 2) 95 3) Page 291
61648	Same	Indirect cursor position cp = 0 for DISPLAY_TEXT and DISPLAY_REG	1) 0 .. 255 2) 0 3) Page 291
61649	Same	Indirect cursor position cp = 0 for USER_INPUT	1) 0 .. 255 2) 0 3) Page 292
61653	Same	Display time for monitor functions in seconds	1) 0 .. 65.535 2) 35 3) Page 292
61683	Same	Text selection for DISPLAY_TEXT_2	1) 0 .. 255 2) 0 3) Page 293
61783	New	Switching over to monitor display	1) 0 .. 1 2) 0 3) Page 293
61787	Same	Indirect Buffer Number with Device # 0	1) 1 .. 4 2) 2 3) Page 285
61788	Same	Multi-Display Mode: Text buffer number for HMI # 1	1) 1 .. 4 2) 1 3) Page 286
61789	Same	Multi-Display Mode: Text buffer number for HMI # 2	1) 1 .. 4 2) 2 3) Page 286
61790	Same	Multi-Display Mode: Text buffer number for HMI # 3	1) 1 .. 4 2) 3 3) Page 286
61791	Same	Multi-Display Mode: Text buffer number for HMI # 4	1) 1 .. 4 2) 4 3) Page 287

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of HMIs (LCD Display) Display			
61792	Same	Multi-Display Mode: Base flag number for HMI # 1	1) -161 .. 1824, 2000 2) 2000 3) Page 287
61793	Same	Multi-Display Mode: Base flag number for HMI # 2	1) -161 .. 1824, 2000 2) 2000 3) Page 287
61794	Same	Multi-Display Mode: Base flag number for HMI # 3	1) -161 .. 1824, 2000 2) 2000 3) Page 288
61795	Same	Multi-Display Mode: Base flag number for HMI # 4	1) -161 .. 1824, 2000 2) 2000 3) Page 288
61796	Same	Multi-Display Mode: Register number for controlling LEDs of HMI # 1	1) 0 .. 20479, 61488 2) 61488 3) Page 289
61797	Same	Multi-Display Mode: Register number for controlling LEDs of HMI # 2	1) 0 .. 20479, 61488 2) 61488 3) Page 289
61798	Same	Multi-Display Mode: Register number for controlling LEDs of HMI # 3	1) 0 .. 20479, 61488 2) 61488 3) Page 289
61799	Same	Multi-Display Mode: Register number for controlling LEDs of HMI # 4	1) 0 .. 20479, 61488 2) 61488 3) Page 290
61818	Same	Max. time for USER_INPUT Default = 0 (disabled)	1) 0 .. 65.535 2) 0 3) Page 294
61825	Same	Number of characters per line	1) 0 .. 255 2) 24 3) Page 294
61826	Same	Number of lines	1) 0 .. 255 2) 2 3) Page 295

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of HMIs (LCD Display) Display			
61827	Same	Number of characters (the product results from the values in registers 61825 and 61826)	1) 0 .. 255 2) 48 3) Page 295

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of HMIs (LCD Display) Restrictions for user input			
61472	Same	Restrictions of monitor functions	1) 0 .. 255 2) 255 3) Page 296

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of HMIs (LCD Display) Mapping of LEDs on HMIs			
61480 .. 61487	Same	Mapping of keys on HMIs	1) 0 .. 255 2) 0 3) Page 297
61488 .. 61489	Same	Mapping of LEDs on HMIs	1) 0 .. 255 2) 0 3) Page 298
61667	Same	Mapping of LEDs on HMIs (Bit 0 ... 11, overlaid by registers 61488 and 61489)	1) 0 .. 65.535 2) 0 3) Page 299

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of HMIs (LCD Display) Baud Rate			
62984	Same	Baud rate of HMI	1) 0 .. 11 2) 6 (9600 Baud) 3) Page 300

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Ethernet control IP-address			
63816	New	IP-address LSB	1) 0 .. 255 2) Stored IP-address LSB 3) Page 177
63817	New	IP address 2SB	1) 0 .. 255 2) Stored IP-address 2SB 3) Page 178
63818	New	IP address 3SB	1) 0 .. 255 2) Stored IP-address 3SB 3) Page 178
63819	New	IP address MSB	1) 0 .. 255 2) Stored IP-address MSB 3) Page 178

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Ethernet control Subnet mask			
63904	New	Subnet mask LSB	1) 0 .. 255 2) Stored subnet mask LSB
63905	New	Subnet mask 2SB	1) 0 .. 255 2) Stored subnet mask 2SB
63906	New	Subnet mask 3SB	1) 0 .. 255 2) Stored subnet mask 3SB
63907	New	Subnet mask MSB	1) 0 .. 255 2) Stored subnet mask MSB

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Ethernet control Gateway			
63900	New	Gateway LSB	1) 0 .. 255 2) Stored gateway LSB
63901	New	Gateway 2SB	1) 0 .. 255 2) Stored gateway 2SB
63902	New	Gateway 3SB	1) 0 .. 255 2) Stored gateway 3SB
63903	New	Gateway MSB	1) 0 .. 255 2) Stored gateway MSB

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Ethernet control Times			
63895	New	Setting timeout time in milliseconds	1) 1 .. 65.535 2) 250 [ms] 3) Page 301
63896	New	Processing time of network instruction in milliseconds	1) 0 .. 65.535 2) 0 3) Page 301

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Ethernet control Error messages			
63897	New	Quantity of network errors	1) 0 .. 255 2) 0 3) Page 302
63898	New	Error code of last network access	1) 0 .. 255 2) 0 3) Page 302

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
		E-Mail	
63827	New	Status of web-function initialization	1) 0 .. 255 2) dependent on initialization 3) Page 303
63828	New	Own IP address	1) 0x00000000 .. 0xffffffff 2) IP-address
63829	New	SMTP server IP address	1) 0x00000000 .. 0xffffffff 2) IP-address
63830	New	POP3 server IP address	1) 0x00000000 .. 0xffffffff 2) IP-address
63831	New	Port number SMTP server	1) 0 .. 65.535 2) Port number
63832	New	Port number POP3 server	1) 0 .. 65.535 2) Port number
63833	New	Processing status of the e-mail	1) 0 .. 255 2) 0 3) Page 304
63834	New	Number of the task which sends an e-mail	1) 0 .. 255 2) 0 3) Page 304

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
RemoteScan function			
63020	New	RemoteScan protocol register	1) 32 bit 2) 0 3) Page 305
63021	New	Number of communication units	1) 32 bit 2) 0 3) Page 305
63022	New	Activity state	1) 32 bit 2) 0 3) Page 305

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Data Files			
63835	New	Processing State	1) 0 .. 255 2) 0 3) Page 306
63836	New	Task number	1) 0 .. 255 2) 255 3) Page 306

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Activation of JETWay Network			
63794	New	Activation of JETWay Network	1) +/- 65.535 2) 0 3) Page 307

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of network 1 Network number, baud rate, indirect addressing			
62995	Same	Network number	1) 0 .. 99 2) 2 3) Page 308
62996	Same	Baud Rate	1) 0 .. 11 2) 11 3) Page 308
61589	Same	Indirect Network Addressing	1) 0 .. 255 2) 0 3) Page 309

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of network 1 Times			
61585	Same	Network response time in milliseconds	1) 0 .. 255 2) Previous time 3) Page 310
61586	Same	Processing time of network instruction	1) 0 .. 255 2) Previous time 3) Page 311
61588	Same	Setting timeout time in milliseconds	1) 0 .. 255 2) 250

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of network 1 Error messages			
61955	Same	Number of checksum errors when receiving via the network	1) -8.388.608 .. +8.388.607 2) 0
63780	New	Error code of last network access	1) 0 .. 255 2) 0 3) Page 311
63781	New	Quantity of network errors	1) 0 .. 255 2) 0 3) Page 311

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of network 2 Configuration			
62989	Same	Configuration network 2	1) 0 .. 255 2) 8 3) Page 312
62990	Same	Baud Rate	1) 0 .. 255 2) 6 3) Page 313

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of network 2 JETWay			
61511	Same	Setting timeout time in milliseconds	1) 0 .. 255 2) 250
62991	Same	Network number	1) 0 .. 255 2) 2 3) Page 313
63773	New	Network response time in milliseconds	1) 0 .. 65.535 2) 0 3) Page 314
63774	New	Quantity of network errors	1) 0 .. 255 2) 0 3) Page 314
63775	New	Error code of last network error	1) 0 .. 255 2) 0 3) Page 315

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of network 2 User-Programmable Interface			
61499	Same	Receive buffer occupancy	1) 0 .. 127 2) 0 3) Page 315
61502	Same	Transmit buffer occupancy	1) 0 .. 127 2) 0 3) Page 315
61508	Same	Interface status	1) 0 .. 127 2) Status 3) Page 316
62992	Same	Transmission Buffer	1) 0 .. 255 2) 0 3) Page 316
62993	Same	Receive buffer with deleting characters	1) 0 .. 255 2) 0 3) Page 317
62994	Same	Receive buffer without deleting characters	1) 0 .. 255 2) 0 3) Page 317

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Control of the Programming Interface (PC)			
62983	Same	Baud rate programming interface	1) 0 .. 255 2) 6 3) Page 318

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Interface Activity Monitoring			
61470	New	Image of flags 2080 through 2087	1) 0 .. 255 2) 0 3) Page 319
61471	New	Image of flags 2088 through 2095	1) 0 .. 255 2) 0 3) Page 320
63980	New	Ethernet Monitoring Time (JetIP)	1) 0 .. 65.535 2) 0 3) Page 322
63981	New	PC monitoring time (pcom5)	1) 0 .. 65.535 2) 0 3) Page 322
63982	New	LCD monitoring time (pcom5)	1) 0 .. 65.535 2) 0 3) Page 322
63983	New	Monitoring time NET 1 (JETWay)	1) 0 .. 65.535 2) 0 3) Page 322
63984	New	Monitoring time NET 2 (JETWay	1) 0 .. 65.535 2) 0 3) Page 323

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Status of the application program			
61440	Same	Status of the application program	1) 0 .. 255 2) Status 3) Page 324
61529	Same	Toggle switch position when switching on	1) 0 .. 255 2) Switch position when switching on 3) Page 324
62976	Same	Current toggle switch position	1) 0 .. 255 2) Switch position 3) Page 325

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Timer register			
63892	New	Start delay	1) 0 .. 300 2) 0 3) Page 326
61645	Same	Number of time registers	1) 0 .. 255 2) 0
61806	Same	User time base in milliseconds Time base of DELAY	1) 0 .. 65.535 2) 100 3) Page 326
61848	Same	Time base for START_TIMER, TIMER_END	1) 0 .. 65.535 2) 61.954

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
General Registers			
61684	Same	Bit code of occupied module slots	1) 0 .. 255 2) Status 3) Page 327
61568 .. 61575	Same	Modules detected in the slots	1) 0 .. 255 2) 0 3) Page 328
61688	Same	Submodule type in socket 1	1) 0 .. 255 2) 99 3) Page 48
61689	Same	Submodule type in socket 2	1) 0 .. 255 2) 99 3) Page 48
61692	New	Submodule type in socket 3	1) 0 .. 255 2) 99 3) Page 48

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Specifying the floating point register array			
61810	New	Specifying the start of the floating point register array	1) 1000 .. 20479 2) 20479 3) Page 329

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
Real-time clock (TRC)			
62912 .. 62919	Same	RTC-buffer, no access to RTC	3) Page 353
62920 .. 62927	Same	RTC direct	3) Page 353
62912, 62920	Same	Seconds	1) 0 .. 255 2) Current time
62913, 62921	Same	Minutes	1) 0 .. 255 2) Current time
62914, 62922	Same	Hours	1) 0 .. 255 2) Current time
62916, 62924	Same	day of the week, 1 = Sunday, 2 = Monday, etc.	1) 0 .. 255 2) Previous setting
62917, 62925	Same	Day	1) 0 .. 255 2) Current date
62918, 62926	Same	Month	1) 0 .. 255 2) Current date
62919, 62927	Same	Year	1) 0 .. 255 2) Current date
62928	New	Dummy write and read access to the real-time clock	1) 0 .. 255 2) 0

Register Number	New / modified / same as D-CPU1	Function	1) Value Range 2) Reset Value 3) Cross Reference
		PC-Card	
61445	Same	PC-card detection	1) 2) 0 3) Page 59
61446	Same	Status message PC-Card	1) 2) 63 3) Page 59
61690	Same	Device ID from tuple 01 device info 1	1) 2) 0 3) Page 59
61691	Same	Device size byte from tuple 01 device info 1	1) 2) 0 3) Page 59

32 Special Registers

32.1 Operating System (Error) Messages

Register 61473: Operating system error	
Function	**Description**
Read	Operating system error
Write	-
Value range	0 to 255 (bit-coded)
Value after reset	0

Meaning of register 61473:

Bit 0 = 0	No timeout when accessing slave register.
Bit 0 = 1	Timeout when accessing slave registers. This bit can also be queried by means of special flag 2104.
Bit 1 = 0	No timeout when accessing slave register since reset.
Bit 1 = 1	Timeout when accessing slave register since reset. This bit can also be queried by means of special flag 2105.
Bit 6 = 0	No timeout during last network access
Bit 6 = 1	Timeout during last network access. This bit can also be queried by means of special flag 2110.
Bit 7 = 0	No network timeout since reset.
Bit 7 = 1	Network timeout since reset. This bit can also be queried by means of special flag 2111.

Register 61477: Operating system error	
Function	**Description**
Read	Operating system error
Write	-
Value range	0 to 255 (bit-coded)
Value after reset	0

The ERR-LED it lit red when the content of register 61477 is unequal zero.

Meaning of register 61477:

Bit 0 = 1	Illegal destination for GOTO or CALL. The task was interrupted. This bit can also be queried by means of special flag 2136.
Bit 1 = 1	Current command would cause stack overflow. The task was interrupted. This bit can also be queried by means of special flag 2137.
Bit 2 = 1	Current command would cause stack underflow. The task was interrupted. This bit can also be queried by means of special flag 2138.
Bit 3 = 1	No application program or CRC error. This bit can also be queried by means of special flag 2139.
Bit 4 = 1	OPC error. This bit can also be queried by means of special flag 2140.
Bit 5 = 1	Syntax error in the interpreter. This bit is set if, for example, a command is to be carried out which is not supported by the interpreter. This bit can also be queried by means of special flag 2141.

Register 61478: Error, operating system messages	
Function	**Description**
Read	Operating system errors and messages
Write	-
Value range	0 to 255 (bit-coded)
Value after reset	0

Meaning of register 61478:

Bit 0 = 1	Error at the digital output. This bit can also be queried by means of special flag 2144.
Bit 1 = 1	Real-time clock found. This bit can also be queried by means of special flag 2145.
Bit 2 = 1	Battery for real-time clock is ok. This bit can also be queried by means of special flag 2146.
Bit 3 = 1	Battery for register-RAM almost empty. This bit can also be queried by means of special flag 2147.

Note!

The real-time clock and the RAM for application registers are backed up by the same battery.

Register 61672: Error message during access to slave modules	
Function	**Description**
Read	Current error message
Write	Illegal
Value range	0 through 255
Value after reset	0

Meaning of register 61672:

Possible error messages:

1 = semaphores
2 = timeout without acknowledgment
3 = incorrect acknowledgment
10 = timeout during access to module bus

Register 63770: Numerator with division by 0	
Function	**Description**
Read	Current result of count
Write	Illegal
Value range	0 through 65,535
Value after reset	0

Meaning of register 63770:

When a division by zero occurs the counter register 63770 is incremented by one. On program start-up this register is initialized with 0.

Register 63771: Program address which has caused division by 0	
Function	**Description**
Read	Program address which has caused division by 0
Write	Illegal
Value range	0 through (256 * 1,024)
Value after reset	0

Meaning of register 63771:

It contains the program address which has caused division by 0.

Register 63787: Current build level	
Function	**Description**
Read	Current build level
Write	Illegal
Value range	0 through 65,535
Value after reset	Current build level

The current build level is a subversion number based on the controller version (register 62977).

32.2 Computing Speed

Register 63796: Computing speed	
Function	**Description**
Read	Last value
Write	0: 100 MHz JC-647 operates at standard speed 1: 133 MHz JC-647 operates at higher speed (by approx. 30 % faster than standard speed).
Value range	0 .. 1
Value after reset	0

32.3 Task Control

Register 61467: Task switch conditions	
Function	**Description**
Read	Last value
Write	New task switch condition
Value range	0 to 255 (bit-coded)
Value after reset	3

Task switch conditions:

A task switch is always carried out when the following conditions are fulfilled:
- DELAY
- USER_INPUT
- WHEN (not fulfilled)

as well as in the following case:
- Flag 2056 AND task switch timeout (reg. 61804)
- Flag 2057 and GOTO
- Flag 2058 and IF (not fulfilled)

The state of flag 2056 is identical with that of bit 0 of register 61467:
Bit 0 = 0: No task switch in case of timeout (reg. 61804)
Bit 0 = 1: Task switch in case of timeout (reg. 61804)

The state of flag 2057 is identical with that of bit 1 of register 61467:
Bit 1 = 0: No task switch in case of GOTO
Bit 1 = 1: Task switch in case of GOTO

The state of flag 2058 is identical with that of bit 2 of register 61467:
Bit 2 = 0: No task switch in case of IF (not fulfilled)
Bit 2 = 1: Task switch in case of IF (not fulfilled)

Register 61474: Multitasking control	
Function	**Description**
Read	Selected interfaces
Write	Add interface
Value range	0 to 255 (bit-coded)
Value after reset	0

Meaning of register 61474:

Bit 0 = 1	The PC interface is selected with each task switch. This bit can also be queried by means of special flag 2112.
Bit 1 = 1	The LCD interface is selected with each task switch. This bit can also be queried by means of special flag 2113.
Bit 2 = 1	The network interface is selected with each task switch. This bit can also be queried by means of special flag 2114.

Registers 185000 .. 185099: Task status	
Function	**Description**
Read	Current task status
Write	Illegal
Value range	0 through 255
Value after reset	Present status

Meaning of registers 185000 .. 185099:

255	Task is being processed
254	DELAY
253	USER_INPUT
250	WHEN_MAX
2	Waiting for response (Ethernet network)
1	TASKBREAK
0	Stopped

Register 185000 indicates the state of task 0, register 185001 indicates the state of task 1, etc. Register 185099 indicates the state of task 99.

32.4 Controlling HMIs (LCDs)

32.4.1 Entering Fixed-Point Numbers

There are four special registers to display fixed-point numbers. These are registers 61452, 61453, and 61786.

Register 61453: Field width (DISPLAY_REG instruction)	
Function	**Description**
Read	Current field width
Write	New field width
Value range	0 through 8
Value after reset	0

Meaning of register 61453:

Defines the number of positions to be displayed. A maximum of 9 positions can be assigned to a register display (8 digits and 1 sign).
If values of two or three digits are to be displayed only, then the actually required number of positions can be assigned to the display by using register 61453. This is of special importance if a great number of texts and values are to be displayed on an HMI.

The following rule applies: Contents of register 61453 = number of positions to be displayed + sign
For example, value of register 61453 = 4 corresponds to 3 positions +1 sign

Note!

It should be considered that the sign and the decimal point occupy one position. If a 6-digit value is to be displayed, then value 7, resp. 8 is to be entered into register 61453.

Register 61781: Sign suppression	
Function	**Description**
Read	Current value for sign suppression.
Write	New value for sign suppression.
Value range	0 through 1
Value after reset	0

Meaning of register 61781:

Register values can be displayed either with or without sign. Values are displayed with sign by default. Using register 61781 allows to suppress the display of signs.

Value 0: Sign will be displayed

Value 1: Sign will not be displayed

Register 61452: Decimal position for DISPLAY_REG	
Function	**Description**
Read	Last value
Write	New value
Value range	0 through 255
Value after reset	0

Meaning of register 61452:

The number of decimal positions is defined through the value of this register. As an alternative, register 61786 can be used instead of register 61452. A maximum of four decimal positions is possible.

Register Value	Decimal Positions
0	0
1	1
2	2
3	3
4	4

Register 61786: Divisor for the value output DISPLAY_REG	
Function	**Description**
Read	Last value
Write	New value
Value range	0 through 65,535
Value after reset	1

Meaning of register 61786:

The number of decimal positions is defined through the value of this register. As an alternative, register 61452 can be used instead of register 61786. A maximum of four decimal positions is possible.

Register Value	Decimal Positions
1	0
10	1
100	2
1.000	3
10.000	4

Example:

REGISTER_LOAD (61786, 100)
or
REGISTER_LOAD (61452, 2)

REGISTER_LOAD (100, 999)
DISPLAY_REG (0, 1, 100)

Value 9.99 is displayed on the LC display.

The numeric value of register 100 remains unchanged. For representation purposes on the display, a decimal point is added only.

32.4.2 Displaying a Suggested Value

Register 61976: Suggested value for USER_INPUT	
Function	**Description**
Read	Existing suggested value
Write	New suggested value
Value range	-8,388,608 through +8,388,607
Value after reset	0

Meaning of register 61976:

By issuing the USER_INPUT instruction, a suggested value can be displayed on the LC-display. The suggested value is written to register 61976. The number of decimal positions is defined in register 61452 or 61786.

Example:

REGISTER_LOAD (61786, 100)
or
REGISTER_LOAD (61452, 2)

REGISTER_LOAD (61976, 555)
USER_INPUT (0, 1, 100)

The suggested value 5.55 is displayed on the LC-display.

32.4.3 Entering Fixed-Point Numbers

There are three special registers to enter fixed-point numbers:registers 61455, 63809, and 63810.

Register 61455: Field width (USER_INPUT instruction)	
Function	**Description**
Read	Current field width
Write	New field width
Value range	0 through 8
Value after reset	8

Meaning of register 61455:

Defines the number of positions to be assigned to the USER_INPUT instruction. A maximum of 8 positions can be assigned to a user input (7 digits and 1 sign).
If values of only two or three digits are to be entered, then the actually required number of positions can be assigned to the display by using register 61455. This is of special importance if a great number of texts and values are to be displayed on an HMI.
It should be considered that the sign occupies one position. If a 6-digit value is to be input, then value 7 must be entered into register 61455.

Register 63809: Divisor for value input USER_INPUT	
Function	**Description**
Read	Last value
Write	New value
Value range	0 through 65,535
Value after reset	1

Meaning of register 63809:

The value of this register desfines the number of decimal positions for data input. The content of register 63809 results from the decimal positions entered via the USER_INPUT. A maximum of four decimal positions is possible.

Register Value	Decimal Positions
1	0
10	1
100	2
1.000	3
10.000	4

Register 63810: Decimal positions for USER_INPUT	
Function	**Description**
Read	Last value
Write	New value
Value range	0 through 255
Value after reset	0

Meaning of register 63810:

The value of this register desfines the number of decimal positions for data input. The content of register 63810 results from the decimal positions entered via the USER_INPUT. A maximum of four decimal positions is possible.

Register Value	Decimal Positions
0	0
1	1
2	2
3	3
4	4

Example:

Data is downloaded from the HMI to register 200 using the following instruction:

```
USER_INPUT (0, 1, 200)
```

Once value `20.00` is entered by the operator, the following values appear in the relevant registers:

Before confirming the USER_INPUT with the ENTER key:

```
Register 200 contains the
previous value

Register 63809 = 100            [Divisor for the value input
                                USER_INPUT]

Register 63810 = 2              [Decimal positions for USER_INPUT]
```

After confirming the USER_INPUT with the ENTER key:

```
Register 200 = 2000

Register 63809 = content of     [Divisor for the value input
register 61786                  USER_INPUT]

Register 63810 = content of     [Decimal positions for USER_INPUT]
register 61452
```

Note!

The numerical value of register 200 is 2000. For representation purposes, a decimal point is added on the display. The operator only has to input the value for register 200 together with the desired decimal places. The values of register 63809 and register 63810 will result from this input.

32.4.4 Indication

Register 61448: Display language	
Function	**Description**
Read	Present setting for the language of integrated HMI functions. Value 0: German Value 1: English
Write	Language selection
Value range	0 through 255
Value after reset	0

Meaning of register 61448:

By using this register, the language for communication functions between HMI and operator is set. The language setting refers to operating system functions of the HMI, not to texts output by the user. Such operating system functions are, for example, the monitor functions for registers, flags, inputs and outputs.

Note!

When loading the program, the register is set to the status of the JetSym dialog language.

Register 61787: Indirect Buffer Number for Device # 0	
Function	**Description**
Read	Set indirect buffer number
Write	New value for indirect buffer number
Value range	1 through 4
Value after reset	2

Meaning of register 61787:

The control system provides four text buffers for multi-display mode. Using the DISPLAY_TEXT or DISPLAY_REG/_VALUE instructions data can be written into this buffer.

When using these instructions, the device number defines the buffer which is activated by the corresponding instruction. If a device number between 1 and 4 is used, the buffer is addressed directly. If device number 0 is used, then that buffer is addressed to which register 61787 points. This way, it is possible to divert a text, for which device number 0 was specified, to several HMIs.

Note!

If only one HMI is connected and if it is configured to run in single mode, this HMI cyclically polls buffer 2 (reset value).

Register 61788: Text buffer for HMI # 1	
Function	**Description**
Read	Set number of text buffer
Write	A new text buffer is assigned to HMI # 1
Value range	1 through 4
Value after reset	1

Register 61789: Text buffer for HMI # 2	
Function	**Description**
Read	Set number of text buffer
Write	A new text buffer is assigned to HMI # 2
Value range	1 through 4
Value after reset	2

Register 61790: Text buffer for HMI # 3	
Function	**Description**
Read	Set number of text buffer
Write	A new text buffer is assigned to HMI # 3
Value range	1 through 4
Value after reset	3

Register 61791: Text buffer for HMI # 4	
Function	**Description**
Read	Set number of text buffer
Write	A new text buffer is assigned to HMI # 4
Value range	1 through 4
Value after reset	4

Meaning of registers 61788 through 61791:

A buffer can be assigned to each display with the help of registers 61788 through 61791.

Register 61792: Basic key flag number for HMI # 1	
Function	**Description**
Read	Set basic number
Write	Basic number of flags which are used for HMI # 1 to recognize keystrokes.
Value range	-161 through 1824, 2000
Value after reset	2000

Register 61793: Basic key flag number for HMI # 2	
Function	**Description**
Read	Set basic number
Write	Basic number of flags which are used for HMI # 2 to recognize keystrokes.
Value range	-161 through 1824, 2000
Value after reset	2000

Register 61794: Basic key flag number for HMI # 3	
Function	**Description**
Read	Set basic number
Write	Basic number of flags which are used for HMI # 3 to recognize keystrokes.
Value range	-161 through 1824, 2000
Value after reset	2000

Register 61795: Basic key flag number for HMI # 4	
Function	**Description**
Read	Set basic number
Write	Basic number of flags which are used for HMI # 4 to recognize keystrokes.
Value range	-161 through 1824, 2000
Value after reset	2000

Meaning of registers 61792 through 61795:

Registers 61792 through 61795 allow to shift the flag area, reflecting the key status of the HMIs, within the whole flag range of the controller.

The flag area for keys is calculated by the following formula:
Flag area for keys = basic number + (160 .. 223)

Note!

The value following a reset maps the keys of all HMIs into the standard flag area for single-display mode, i.e. from flag 2160 through 2223.

Example:
Following a reset, the F1 key is mapped to flag 2201 since the basic number is 2000.

Register 61796: Register number for controlling LEDs of HMI # 1	
Function	**Description**
Read	Set register number for controlling LEDs of HMI # 1
Write	Defining a register the bits of which control the LEDs of HMI # 1.
Value range	0 through 20479, 61488
Value after reset	61488

Register 61797: Register number for controlling LEDs of HMI # 2	
Function	**Description**
Read	Set register number for controlling LEDs of HMI # 2
Write	Defining a register the bits of which control the LEDs of HMI # 2.
Value range	0 through 20479, 61488
Value after reset	61488

Register 61798: Register number for controlling LEDs of HMI # 3	
Function	**Description**
Read	Set register number for controlling LEDs of HMI # 3
Write	Defining a register the bits of which control the LEDs of HMI # 3.
Value range	0 through 20479, 61488
Value after reset	61488

Register 61799: Register number for controlling LEDs of HMI # 4	
Function	**Description**
Read	Set register number for controlling LEDs of HMI # 4
Write	Defining a register the bits of which control the LEDs of HMI # 4.
Value range	0 through 20479, 61488
Value after reset	61488

Meaning of registers 61796 through 61799:

These registers are for assigning flags, which control LEDs of HMIs, to several address areas. Then, LEDs can then be addressed in a different way using the same key on several HMIs.

Following a reset the LEDs of all HMIs are assigned to those flags to which they are assigned in single-display mode, i.e. to the flags 2224 through 2235.

Now, a register can be assigned to each HMI with the help of registers 61796 through 61799. The lower 12 bits of these registers, then, control the LEDs.
If a given register is overlaid by flags, LEDs can also be addressed via these flags and not via register bits.

Example:
Flags 2224 through 2239 are overlaid on register 61488.

Register 61461: "Delete End Of Line" character	
Function	**Description**
Read	Latest set value
Write	New value
Value range	0 through 255
Value after reset	36 (ASCII code of “$“)

Meaning of register 61461:

This register contains the ASCII code of the DELEOL (Delete End Of Line) character. After a reset, the register contains value 36 which is the ASCII code of “$“ (dollar sign). If you want to display a dollar sign in a text, this register can be used to transfer the function of “$“ to another character by entering another number.

Register 61462: "Delete Screen" character	
Function	**Description**
Read	Latest set value
Write	New value
Value range	0 through 255
Value after reset	95 (ASCII code of "_")

Meaning of register 61462:

This register contains the ASCII code of the DELSCR (DELete SCReen) character. This characters clears the LC-display. After a reset, the register contains value 95 which is the ASCII code of "_" (underscore). This character can be replaced by any other character.

Example:

```
REGISTER_LOAD (61462, 38)
DISPLAY_TEXT (0, 0, "&")
```

These two instructions clear the display. The DELSCR character is first modified and then immediately be used in the DISPLAY_TEXT instruction.

Register 61648: Indirect Cursor Position cp=0 for DISPLAY_TEXT and DISPLAY_REG	
Function	**Description**
Read	Set cursor position
Write	New cursor position
Value range	0 through 255
Value after reset	0

Meaning of register 61648:

The value of this register indicates the cursor position for the instructions DISPLAY_TEXT or DISPLAY_REG if cursor position 0 was specified in the instruction.
If the value of this register is 0 as well, which is the case after a reset, then the text to be displayed is simply added to the last line.

Register 61649: Indirect Cursor Position cp=0 for USER_INPUT	
Function	**Description**
Read	Set cursor position
Write	New cursor position
Value range	0 through 255
Value after reset	0

Meaning of register 61649:

This register has exactly the same function as the above mentioned but for the USER_INPUT instruction. It determines the cursor position where the entry takes place when cursor position 0 was entered for user input.
This value is also 0 after a reset. Therefore, the text is added to the last input or output.

Register 61653: Display time for monitor functions	
Function	**Description**
Read	Set display time in seconds
Write	Newly selected display time in seconds
Value range	0 through 65,535
Value after reset	35

Meaning of register 61653:

The display of HMIs works with two display levels. Usually, level 1 is displayed. This is the level that is also accessed by the application program. This level displays the instructions DISPLAY_TEXT, DISPLAY_REG and USER_INPUT.
Registers, flags, inputs and outputs can be queried using the user inteface by pressing the keys “R“ and “I/O“. This takes place on display level 2 (so-called monitor screen).

The time in seconds after which switching back to level 1 is to take place, can be defined in register 61653.
After a reset, this register has the value 35, i.e. 35 seconds.

Register 61683: Text selection for DISPLAY_TEXT_2	
Function	**Description**
Read	Current value to select text 1 or text 2 in conjunction with the DISPLAY_TEXT_2 instruction
Write	New value for the selection of text 1 or text 2
Value range	0 through 255
Value after reset	0

Meaning of register 61683:

Whether text 1 or text 2 is to be displayed on the HMI can be selected by means of register 61683. This is, for example, useful for bilingual operator guidance. text 1 for the customer and text 2 for the service staff.
The display is realized using the following instruction:

```
DISPLAY_TEXT_2 (0, 1, "Text 1", "Text 2")
```

Register 61683	Displayed Text
0	Text 1
1	Text 2

Register 61783: Switching over to monitor display	
Function	**Description**
Read	Present state: Direct switchover to the monitor screen can be carried out by pressing the ENTER key.
Write	New state for switchover to monitor screen mode
Value range	0 through 1
Value after reset	0

Meaning of register 61783:

By pressing the *ENTER* key, direct switching over to monitor screen can be carried out. This function can be enabled or disabled by using register 61783.

Value 0: Switching over by pressing *ENTER* enabled
Value 1: Switching over by pressing *ENTER* disabled

Register 61818: Max. time for USER_INPUT	
Function	**Description**
Read	Current set value for the time
Write	New value for the time after which USER_INPUT is to be terminated
Value range	0 through 65,535
Value after reset	0 (disabled)

Meaning of register 61818:

The operating system knows from this register how much time (in seconds) the user has for an input using the USER_INPUT.
After a reset, this register has the value 0, which means: there is no time limit for the user to make the entry and confirm it with ENTER. The number is then read and the program is continued to being processed. Setting the register, for example, to 10 will terminate the USER_INPUT instruction after 10 seconds.

Meaning of flag 2053:

This flag is to be seen in conjunction with the special register "max. time for USER_INPUT". After a USER_INPUT, this flag can be queried whether the user has terminated the input correctly or whether the user input was terminated by a timeout. Flag 2053 = 1 means the user input was terminated by a timeout.

Register 61825: Number of characters per line	
Function	**Description**
Read	Number of character per line of the connected display
Write	-
Value range	0 through 255
Value after reset	24 (if no display is connected)

Meaning of register 61825:

This register indicates the number of characters per line on the connected display. If no display is connected, value 24 will be displayed.

Register 61826: Number of lines	
Function	**Description**
Read	Number of lines on the connected display
Write	-
Value range	0 through 255
Value after reset	2 (if no display is connected)

Meaning of register 61826:

This register indicates the number of lines on the connected display. If no display is connected, value 2 will be displayed.

Register 61827: Number of characters	
Function	**Description**
Read	Number of characters on the connected display
Write	-
Value range	0 through 255
Value after reset	48 (if no display is connected)

Meaning of register 61827:

This register indicates the number of characters on the connected display. The number of characters is the product of values in registers 61825 and 61826. If no display is connected, value 48 is displayed.

32.4.5 Restrictions for User Input

Register 61472: Restriction of monitor functions	
Function	**Description**
Read	Status of monitor functions
Write	Disabling/enabling monitor functions
Value range	0 to 255 (bit-coded)
Value after reset	255

Meaning of register 61472:

Flags 2096 (bit 0) to 2103 (bit 7) are overlaid by this register.

Monitor functions can be enabled or disabled by this register to allow the user to have access to defined operating functions.
If monitor functions disabled for service staff are to be enabled again, this register can be used as well.

0 = function disabled, 1 = function enabled

Register Bit 61472	Function
Bit 0 = 1 Bit 0 = 0	Display a register by pressing the "R" key No function is assigned to the "R" key
Bit 1 = 1 Bit 1 = 0	Display a flag (2 times "R") Flags cannot be displayed
Bit 2 = 1 Bit 2 = 0	Displaying outputs by pressing the “I/O“ key Outputs cannot be displayed
Bit 3 = 1 Bit 3 = 0	Displaying inputs (2 times “I/O“) Inputs cannot be displayed
Bit 4 = 1 Bit 4 = 0	Modifiying register contents by pressing the "=" key Register contents cannot be modified
Bit 5 = 1 Bit 5 = 0	Modifiying states of flags by pressing the "=" key Flag states cannot be modified
Bit 6 = 1 Bit 6 = 0	Modifiying outputs states by pressing the "=" key Outputs states cannot be modified
Bit 7 = 1 Bit 7 = 0	Permanent display of input by pressing the “=“ key Function disabled

32.4.6 Mapping of LEDs on HMIs

Registers 61480 through 61487: Mapping of keys on HMIs	
Function	**Description**
Read	Current mapping of keys pressed on the HMI
Write	-
Value range	0 to 255 (bit-coded)
Value after reset	0

Meaning of registers 61480 through 61487:

These registers map the keys pressed on the HMI, or the flags of special flags 2160 through 2223.

The following assignment applies:

Overlaying of special flags for register 61480								
Bit number	0	1	2	3	4	5	6	7
Flags	2160	2161	2162	2163	2164	2165	2166	2167

Overlaying of special flags for register 61481								
Bit number	0	1	2	3	4	5	6	7
Flags	2168	2169	2170	2171	2172	2173	2174	2175

Overlaying of special flags for register 61482								
Bit number	0	1	2	3	4	5	6	7
Flags	2176	2177	2178	2179	2180	2181	2182	2183

Overlaying of special flags for register 61483								
Bit number	0	1	2	3	4	5	6	7
Flags	2184	2185	2186	2187	2188	2189	2190	2191

Overlaying of special flags for register 61484								
Bit number	0	1	2	3	4	5	6	7
Flags	2192	2193	2194	2195	2196	2197	2198	2199

Overlaying of special flags for register 61485								
Bit number	0	1	2	3	4	5	6	7
Flags	2200	2201	2202	2203	2204	2205	2206	2207

Overlaying of special flags for register 61486								
Bit number	0	1	2	3	4	5	6	7
Flags	2208	2209	2210	2211	2212	2213	2214	2215

Overlaying of special flags for register 61487								
Bit number	0	1	2	3	4	5	6	7
Flags	2216	2217	2218	2219	2220	2221	2222	2223

Registers 61488 through 61489: Mapping of LEDs on the HMI keys	
Function	**Description**
Read	Current mapping of LEDs on the HMI keys
Write	-
Value range	0 to 255 (bit-coded)
Value after reset	0

Meaning of registers 61488 and 61489:

These registers map the LEDs on the HMI keys, or the flags of special flags 2224 through 2235, respectively.

The following assignment applies:

Overlaying of special flags for register 61488								
Bit number	0	1	2	3	4	5	6	7
Flags	2224	2225	2226	2227	2228	2229	2230	2231

Overlaying of special flags for register 61489								
Bit number	0	1	2	3	4	5	6	7
Flags	2232	2233	2234	2235	-	-	-	-

Register 61667: Mapping of LEDs on the HMI keys	
Function	**Description**
Read	Current mapping of LEDs on the HMI keys
Write	-
Value range	0 to 65,535 (bit-coded)
Value after reset	0

Meaning of register 61667:

This register maps the LEDs on the HMI keys, or the flags of special flags 2224 through 2235, respectively.

The following assignment applies:

Overlaying of user flags for register 61667								
Bit number	0	1	2	3	...	9	10	11
Flags	2224	2225	2226	2227	...	2233	2234	2235

32.4.7 Baud rate of HMI (LCD)

Register 62984: Baud rate of HMI (LCD)	
Function	**Description**
Read	Current baud rate
Write	New baud rate
Value range	0 through 11
Value after reset	6 (9600 baud)

Meaning of register 62984:

Setting	Baud rate in baud (bit/s)
0	150
1	300
2	600
3	1,200
4	2,400
5	4,800
6	9,600 (default)
7	19,200
8	38,400
10	76,800
11	115,200

32.5 Ethernet Control

32.5.1 Addressing

For assigning the IP-address, please refer to chapter 21.1 "Setting the IP-Address" on page 177.

32.5.2 Times

Register 63895: Time for timeout	
Function	**Description**
Read	Setting timeout time in milliseconds
Write	Illegal
Value range	0 through 65,535
Value after reset	250 [ms]

Meaning of register 63895:

Indicates the time after which the communication attempt is terminated with the message "timeout". The time is indicated in milliseconds.

Register 63896: Processing time of network instruction	
Function	**Description**
Read	Processing time in milliseconds
Write	Illegal
Value range	0 through 65,535
Value after reset	0

Meaning of register 63896:

Indicates the processing time of the last executed network command. The processing time comprises the cycle times of both controllers as well as transmission times.

32.5.3 Error Messages

Register 63898: Error code of last network access	
Function	**Description**
Read	Current error code
Write	Illegal
Value range	0 through 255
Value after reset	0

Meaning of register 63898:

Current error code:

0 = No error
1 = Timeout
3 = Error message of the remote station

Register 63897: Quantity of network errors	
Function	**Description**
Read	Current number of errors
Write	Only clearing makes sense
Value range	0 through 255
Value after reset	0

32.6 Web-Function Initialization

Register 63827: State of Web function Initialization	
Function	**Description**
Read	Present status
Write	Illegal
Value range	0 .. 255
Value after reset	Depends on initialization

Meaning of register 63827:

Bit 0 = 0:	FTP server not available
Bit 0 = 1:	FTP server available
Bit 1 = 0:	HTTP-server not available
Bit 1 = 1:	HTTP-server available
Bit 2 = 0:	E-mail function not available
Bit 2 = 1:	E-mail available
Bit 3 = 0:	Data file feature not available
Bit 3 = 1:	Data file feature available
Bit 4 = 0:	No Modbus/TCP
Bit 4 = 1:	Modbus/TCP available
Bit 5 = 0:	Modbus/TCP server not available
Bit 5 = 1:	Modbus/TCP Server has been started
Bit 13 = 0:	E-mail has been sent
Bit 13 = 1:	E-mail is being sent

32.7 E-mail Tunction

The status of e-mail processing can be queried from the following two registers:

Register 63833: Processing status of the e-mail	
Function	**Description**
Read	Current processing status
Write	Illegal
Value range	0 .. 255
Value after reset	0

Meaning of register 63833:

Current processing status:

0 = e-mail not processed.
1 = transfer to e-mail module is being executed.
2 = e-mail is being created.
3 = e-mail was sent to the server.

Register 63834: Task number	
Function	**Description**
Read	Number of the task sending an e-mail
Write	Illegal
Value range	0 .. 255
Value after reset	0

Meaning of register 63834:

0 .. 99: Number of the task sending an e-mail.
255: No task is sending an e-mail

Note!

Evaluating register 63834 is only useful when the value in register 63833 is not 0.

32.8 RemoteScan Function

The following registers provide an overview of the set RemoteScan mode.

Register 63020: RemoteScan protocol register	
Function	**Description**
Read	Set type of protocol
Write	Illegal
Value range	32 bits
Value after reset	0

Meaning of register 63020:

Set protocol of the RemoteScan function:

5 = Modbus/TCP

Register 63021: Amount of communication units	
Function	**Description**
Read	Amount of communication units
Write	Illegal
Value range	32 bits
Value after reset	0

Register 63022: Activity state	
Function	**Description**
Read	0 = not active, stopped 1 = active (running)
Write	Illegal
Value range	32 bits
Value after reset	0

32.9 Data Files

The processing state of a file operation and the number of the task carrying out the operation can be read from two registers.

Register 63835: Processingtate	
Function	**Description**
Read	Current processing state
Write	Illegal
Value range	0 .. 255
Value after reset	0

Meaning of register 63835:

Current processing state:

0 = No file operation in progress.
1 = Processing transferred to file module.
2 = Data are being read/written
3 = File operation completed

Register 63836: Task number	
Function	**Description**
Read	Number of the task performing a file operation
Write	Illegal
Value range	0 .. 255
Value after reset	255

Meaning of register 63836:

This register indicates which task is performing a file operation:

0 .. 99:	Task number
255:	No task

32.10 Instruction Register

Register 63794: Command Register	
Function	**Description**
Read	Last command
Write	New instruction
Value range	10, 11, 13, 102, 107
Value after reset	0

Meaning of register 63794:

Commands:

10 Activating the four counters

Digital input 101 is connected with counter 1.
Digital input 102 is connected with counter 2.
Digital input 103 is connected with counter 3.
Digital input 104 is connected with counter 4.

11 Deactivating the four counters

Digital inputs 101 through 104 are digital inputs of the PLC.

13 Activation of JETWay Network

This command must no longer be issued.
Reason:
The network commands N_GET_REGISTER, or N_GET_VALUE, and N_SEND_REGISTER, or N_SEND_VALUE will no longer be used with Ethernet. Thus, switching between Ethernet and JETWay network is no longer required. By default, the JETWay network will be activated following a reset.

102 Triggers a software reset of the controller

107 Causes the flash disk to be formatted

32.11 Controlling Network # 1

32.11.1 Address, baud rate

Register 62995: Network number of network # 1	
Function	**Description**
Read	Current node number on the JETWay network
Write	New node number on the JETWay network
Value range	0 through 99
Value after reset	2

Meaning of register 62995:

Setting	Meaning
0	Inactive
1	Master in the JETWay
2 .. 99	Possible node number as slave in JETWay

A network number is assigned to each controller connected to network 1 (node). This network number is written to register 62995.

Register 62996: Baud rate of network # 1	
Function	**Description**
Read	Current baud rate in the JETWay
Write	New baud rate in the JETWay
Value range	0 through 11
Value after reset	11

Meaning of register 62996:

Setting	Baud rate in baud (bit/s)
0	150
1	300
2	600
3	1,200
4	2,400
5	4,800
6	9,600
7	19,200
8	38,400
10	76,800
11	115,200 (default)

Register 61589: Indirect network addressing	
Function	**Description**
Read	Current indirect network number
Write	New indirect network number
Value range	0 through 255
Value after reset	0

Meaning of register 61589:

The content of a source register of a master controller is to be transferred into the destination register of several slave controllers (register number is the same). In this case, indirect network addressing is required.

In the following example, the content of register 100 of the master controller is copied into register 500 of ten slave controllers with network numbers 2 through 11. Indirect addressing is used in example a.), and direct addressing in example b.).

Example:

a.)

```
                .
                .
                .
REGISTER_LOAD (61589, 2)

LABEL Next_Slave
N_SEND_REGISTER (0, 100, 500)
REGINC 61589

IF
    REG 61589
    <
    12
THEN
    GOTO Next_Slave

THEN
                .
                .
                .
```

b.)

```
                .
                .
                .
N_SEND_REGISTER (2, 100, 500)
N_SEND_REGISTER (3, 100, 500)
N_SEND_REGISTER (4, 100, 500)
N_SEND_REGISTER (5, 100, 500)
N_SEND_REGISTER (6, 100, 500)
N_SEND_REGISTER (7, 100, 500)
N_SEND_REGISTER (8, 100, 500)
N_SEND_REGISTER (9, 100, 500)
N_SEND_REGISTER (10, 100, 500)
N_SEND_REGISTER (11, 100, 500)
                .
                .
                .
```

32.11.2 Times

Register 61585: Network response time	
Function	**Description**
Read	Response time in milliseconds
Write	Illegal
Value range	0 through 255
Value after reset	Previous time

Meaning of register 61585:

The time starts upon a network command and ends when the slave controller has received the response.

Register 61586: Processing time of network instruction	
Function	**Description**
Read	Processing time in milliseconds
Write	Illegal
Value range	0 through 255
Value after reset	Previous time

Meaning of register 61586:

Indicates the processing time of the last executed network command. The processing time comprises the network response time plus the time necessary to process the response from the slave controller.

32.11.3 Error Messages

Register 63780: Error code of last network access	
Function	**Description**
Read	Current error code
Write	Illegal
Value range	0 through 255
Value after reset	0

Meaning of register 63780:

Current error code:
0 = No error
1 = Timeout
2 = checksum error
3 = error message from slave
4 = no master specified

Register 63781: Quantity of network errors	
Function	**Description**
Read	Current number of errors
Write	Illegal
Value range	0 through 255
Value after reset	0

32.12 Controlling Network # 2

32.12.1 Configuration of network # 2

Register 62989: Configuration of network # 2	
Function	**Description**
Read	Present configuration
Write	New configuration
Value range	0 through 255
Value after reset	8 (PRIM, RS-232, 8 data bits, 1 stop bit, no parity, 9600 baud)

Meaning of register 62989:

Possible configurations	
Bit 0	0 = user programmable interface (PRIM) 1 = JETWay
Bit 1	Reserved
Bit 2	0 = 1 stop bit 1 = 2 stop bits
Bit 3	0 = 7 data bits per character 1 = 8 data bits per character
Bit 5.4	00 = no parity 01 = illegal 10 = odd parity 11 = even parity
Bit 7.6	00 = RS-232 01 = RS-485 (4-wire) 10 = RS-422 11 = RS-485 (2-wire)

Note!

When bit 0 is set, network operation (JETWay) is stopped. The other bits are ignored in this case.

Full duplex mode is not supported by RS-485 (4-wire).

Register 62990: Baud rate network 2	
Function	**Description**
Read	Current baud rate
Write	New baud rate
Value range	0 through 11
Value after reset	6

Meaning of register 62990:

Setting	Baud rate in baud (bit/s)
0	150
1	300
2	600
3	1,200
4	2,400
5	4,800
6	9,600 (default)
7	19,200
8	38,400
10	76,800
11	115,200

32.12.2 JETWay

Register 62991: Network number of network # 2	
Function	**Description**
Read	Current node number on the JETWay network
Write	New node number on the JETWay network
Value range	0 through 99
Value after reset	2

Meaning of register 62991:

Setting	Meaning
0	Inactive
1	Master in the JETWay
2 .. 99	Possible node number as slave in JETWay

A network number is assigned to each controller connected to network 2 (node). This network number is written to register 62991.

Register 63773: Network response time	
Function	**Description**
Read	Response time in milliseconds
Write	Illegal
Value range	0 through 65,535
Value after reset	Previous time

Meaning of register 63773:

The time starts upon a network command and ends when the slave controller has received the response.

Register 63774: Quantity of network errors	
Function	**Description**
Read	Current number of errors
Write	Illegal
Value range	0 through 255
Value after reset	0

Register 63775: Error code of last network error	
Function	**Description**
Read	Current error code
Write	Illegal
Value range	0 through 255
Value after reset	0

Meaning of register 63775:

Current error code:

0 = No error
1 = Timeout
2 = checksum error
3 = error message from slave
4 = no master specified

32.12.3 User-Programmable Interface

Register 61499: Receive Buffer Occupancy	
Function	**Description**
Read	Current receive buffer occupancy
Write	Illegal
Value range	0 through 127
Value after reset	0

Register 61502: Transmit Buffer Occupancy	
Function	**Description**
Read	Current transmit buffer occupancy
Write	Illegal
Value range	0 through 127
Value after reset	0

Register 61508: Interface status	
Function	**Description**
Read	Present interface status
Write	Illegal
Value range	0 through 127
Value after reset	Status

Meaning of register 61508:

Possible interface status:

Bit 0 = 1 One or more characters lost while receiving

Bit 1 = 1 Stop bit error

Bit 2 = 1 Parity error

Bit 3 = 1 Bits 0 through 2 are or were unequal 0

Bit 4 = 1 Reveive buffer overflow

Bit 5 = 1 Transmit buffer overflow

Register 62992: Transmission Buffer	
Function	**Description**
Read	Last character sent
Write	Send character
Value range	0 through 255
Value after reset	0

Note!

Transmit buffer size is 127 characters

Register 62993: Reception buffer Access removes character	
Function	**Description**
Read	Received character
Write	Illegal
Value range	0 through 255
Value after reset	0

Register 62994: Reception buffer Access does not remove character	
Function	**Description**
Read	Received character
Write	Illegal
Value range	0 through 255
Value after reset	0

Note!

Receive buffer size 127 characters

32.13 Control of the Programming Interface (PC)

Register 62983: Baud rate of programming interface	
Function	**Description**
Read	Current baud rate
Write	New baud rate
Value range	0 through 11
Value after reset	6

Meaning of register 62983:

Setting	Baud rate in baud (bit/s)
0	150
1	300
2	600
3	1,200
4	2,400
5	4,800
6	9,600 (default)
7	19,200
8	38,400
10	76,800
11	115,200

32.14 Interface Activity Monitoring

The activity of a communication partner can be monitored from the application program by means of two flags per interface. This allows to determine, for instance, whether a connection to the display still exists.

Upon receipt of a valid frame the first flag is set by the operating system. Once the first flag is set, the user can set a second flag.

Each time a valid frame is received, a new monitoring interval is started. On termination of the monitoring interval, both flags are reset.
Since the second flag is not set by the OS, the user can detect that the connection was disconnected. Even if the first flag was reset only for a short time.
The monitoring interval can be defined by the user through a register for each interface.

Register 61470: Interface Activity Monitoring Part 1	
Function	**Description**
Read	Present state of the interface activity
Write	Bit 1 can be set
Value range	0 through 255
Value after reset	0

Meaning of register 61470:

Bit 0 = 0	No JETWay activity on the NET2 interface. This bit is reset by the OS once there is no activity on this interface. This bit can also be queried by means of special flag 2080.
Bit 0 = 1	JETWay activity on the interface NET 2. This bit is set by the OS once a valid frame has been received. Then, monitoring interval NET 2 is restarted. This bit can also be queried by means of special flag 2080.
Bit 1 = 0	No JETWay activity on the NET2 interface. This bit is reset together with bit 0 once monitoring interval NET 2 has elapsed. This bit can also be queried by means of special flag 2081.
Bit 1 = 1	This bit can be set by the application program. This bit can be set if bit 0 (flag 2080) has been set. This bit can also be queried by means of special flag 2081.

Register 61471: Interface Activity Monitoring Part 2	
Function	**Description**
Read	Present state of the interface activity
Write	Bit 1, 3, 5, and 7 can be set
Value range	0 through 255
Value after reset	0

Meaning of register 61470:

Bit 0 = 0 — No JETIP activity on the Ethernet interface.
This bit is reset by the OS once there is no activity on this interface
This bit can also be queried by means of special flag 2088.

Bit 0 = 1 — JETIP activity on the Ethernet interface.
This bit is set by the OS once a valid frame has been received.
Then, monitoring interval Ethernet is restarted.
This bit can also be queried by means of special flag 2088.

Bit 1 = 0 — No JETIP activity on the Ethernet interface.
This bit is reset together with bit 0 once monitoring interval Ethernet has elapsed.
This bit can also be queried by means of special flag 2089.

Bit 1 = 1 — This bit can be set by the application program.
This bit can be set if bit 0 (flag 2089) has been set.
This bit can also be queried by means of special flag 2089.

Bit 2 = 0 — No activity on the PC interface.
This bit is reset by the OS once there is no activity on this interface.
This bit can also be queried by means of special flag 2090.

Bit 2 = 1 — Activity on the PC interface.
This bit is set by the OS once a valid frame has been received.
Then, monitoring interval PC is restarted.
This bit can also be queried by means of special flag 2090.

Bit 3 = 0 — No activity on the PC interface.
This bit is reset together with bit 2 once monitoring interval PC has elapsed.
This bit can also be queried by means of special flag 2091.

Bit 3 = 1 — This bit can be set by the application program.
This bit can be set if bit 2 (flag 2091) has been set.
This bit can also be queried by means of special flag 2091.

Bit 4 = 0 — No activity on the LCD interface.
This bit is reset by the OS once there is no activity on this interface.
This bit can also be queried by means of special flag 2092.

Bit 4 = 1	Activity on the LCD interface. This bit is set by the OS once a valid frame has been received. Then, monitoring interval LCD is restarted. This bit can also be queried by means of special flag 2092.
Bit 5 = 0	No activity on the LCD interface. This bit is reset together with bit 4 once monitoring interval LCD has elapsed. This bit can also be queried by means of special flag 2093.
Bit 5 = 1	This bit can be set by the application program. This bit can be set if bit 4 (flag 2093) has been set. This bit can also be queried by means of special flag 2093.
Bit 6 = 0	No JETWay activity on the NET1 interface. This bit is reset by the OS once there is no activity on this interface. This bit can also be queried by means of special flag 2094.
Bit 6 = 1	JETWay activity on the NET1 interface. This bit is set by the OS once a valid frame has been received. Then, the NET1 monitoring interval is restarted. This bit can also be queried by means of special flag 2094.
Bit 7 = 0	No JETWay activity on the NET1 interface. This bit is reset together with bit 6 once monitoring interval NET 1 has elapsed. This bit can also be queried by means of special flag 2095.
Bit 7 = 1	This bit can be set by the application program. This bit can be set if bit 6 (flag 2095) has been set. This bit can also be queried by means of special flag 2095.

Note!

The interface activity monitoring function can only be used from the server. Since only the server is able to determine whether it can receive data from a client.

With visualization systems (e.g. VIADUKT, JetLink, browsers with Java applets) or the programming system JetSym, the scan time can be set or communication can be stopped completely. This fact has to be taken into account when using activity monitoring.

Activity monitoring requires, for instance, cyclical queries to be sent to the control by pcom5, JetIP, or JETWay.

Register 63980: Ethernet monitoring interval (JetIP)	
Function	**Description**
Read	Current monitoring interval in milliseconds
Write	New monitoring interval in milliseconds
Value range	0 through 65,535
Value after reset	0 (no monitoring)

Register 63981: PC monitoring interval (pcom5)	
Function	**Description**
Read	Current monitoring interval in milliseconds
Write	New monitoring interval in milliseconds
Value range	0 through 65,535
Value after reset	0 (no monitoring)

Register 63982: LCD monitoring interval (pcom5)	
Function	**Description**
Read	Current monitoring interval in milliseconds
Write	New monitoring interval in milliseconds
Value range	0 through 65,535
Value after reset	0 (no monitoring)

Register 63983: monitoring interval NET 1 (JETWay)	
Function	**Description**
Read	Current monitoring interval in milliseconds
Write	New monitoring interval in milliseconds
Value range	0 through 65,535
Value after reset	0 (no monitoring)

Register 63984: monitoring interval NET 2 (JETWay)	
Function	**Description**
Read	Current monitoring interval in milliseconds
Write	New monitoring interval in milliseconds
Value range	0 through 65,535
Value after reset	0 (no monitoring)

Note!

Interface activity monitoring can be switched off by setting the monitoring interval to "0".

For HMIs (e.g. LCD 9, LCD 34, LCD 19, etc.) the monitoring interval should not be set to a value less than 200 ms.
Otherwise it may happen that in multi-display mode or with extensive messages errors will be reported, although the communication is still active.

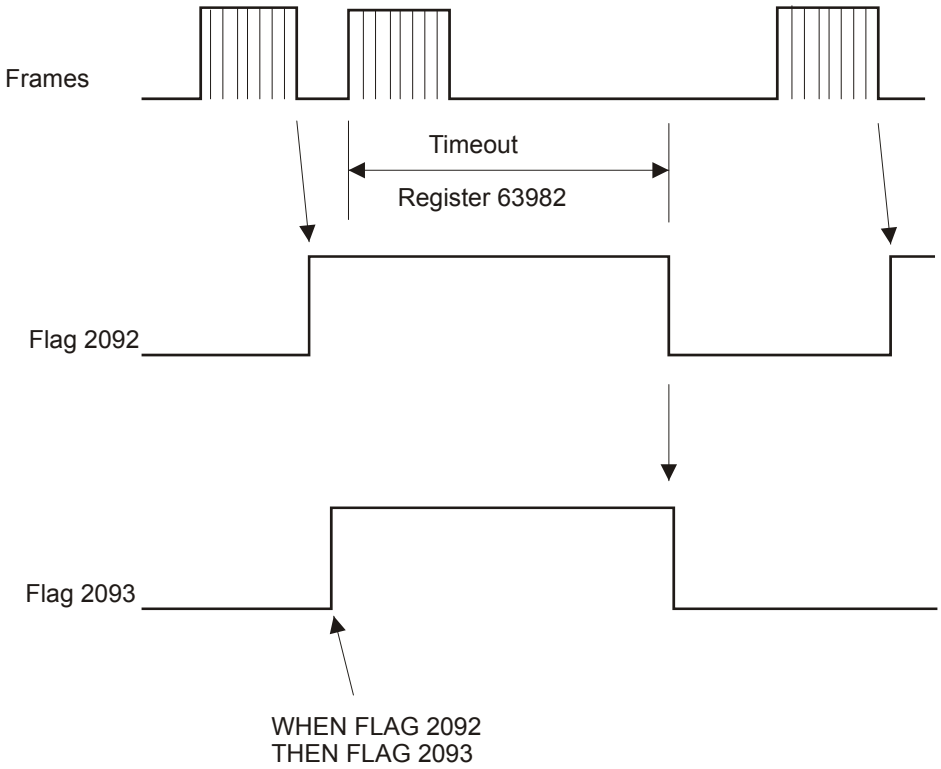


Fig. 54: Monitoring of the LCD interface

32.15 Status of the Application Program

Register 61440: Status of application program	
Function	**Description**
Read	Present status of application program
Write	Illegal
Value range	0 through 255
Value after reset	Status

Meaning of register 61440:

Possible status of the application program:

Bit 0 = 0 Application program stopped

Bit 0 = 1 Application program is running

Register 61529: Toggle switch position when switching on	
Function	**Description**
Read	Toggle switch position when switching on
Write	Illegal
Value range	0 through 255
Value after reset	Toggle switch position when switching on

Meaning of register 61529:

Possible toggle switch positions:

0 = Stop
1 = Load
2 = Run

Register 62976: Current toggle switch position	
Function	**Description**
Read	Current toggle switch position
Write	Illegal
Value range	0 through 255
Value after reset	Toggle switch position when switching on

Meaning of register 62976:

Possible toggle switch positions:

0 = Stop
1 = Load
2 = Run

32.16 Timer Registers

Register 63892: Start delay	
Function	**Description**
Read	Currently set value
Write	Set delay time
Value range	0 through 300
Value after reset	0

Meaning of register 63892:

When the controller is powered-on, i.e. is supplied with a voltage of 24 V, then a delay time is running. This time can be specified in register 63892. The modules at the system bus are initialized and the application program is started only after expiry of this time.
Only the red LED (ERR) is lit while this time is elapsing.

The value in register 63892 specified the delay time in multiples of 100 ms.
The min. value is 0 (function disabled). The max. value is 300, i.e. 30 seconds.

Register 61806: User time base	
Function	**Description**
Read	Set time in milliseconds
Write	New time in milliseconds
Value range	0 through 65,535
Value after reset	100

Meaning of register 61806:

Register 61806 is the time base of the DELAY instruction. By default, 100 ms are set as time base. Thus, DELAY 10 corresponds to one second.

Note!

Very small time units mean additional handling effort for the operating system.

32.17 General Registers

Register 61684: Bit code of occupied module slots	
Function	**Description**
Read	Current bit code
Write	Illegal
Value range	0 through 255
Value after reset	Current bit code

Meaning of register 61684:

Register 61684 informs whether slave modules are plugged and if yes, in which slot. A slave module is, for example, a JX6-CON1 or a JX6-IO64 module. Slave modules are modules plugged in slots 2 through 8 of the controller.

Bit 0 = 0	No module plugged in module slots 2 through 8
Bit 0 = 1	At least one module plugged in module slots 2 through 8
Bit 1 = 1	Module plugged in module slot 2
Bit 2 = 1	Module plugged in module slot 3
Bit 3 = 1	Module plugged in module slot 4
Bit 4 = 1	Module plugged in module slot 5
Bit 5 = 1	Module plugged in module slot 6
Bit 6 = 1	Module plugged in module slot 7
Bit 7 = 1	Module plugged in module slot 8

Register 61568 .. 61575: Detected module	
Function	**Description**
Read	ID for the detected module
Write	Illegal
Value range	0 through 255
Value after reset	Current bit code

Meaning of register 61568 .. 61575:

If a module is plugged in module slot 2, the ID of this module is read from register 61568.

The JX6-IO64 module has a value as ID smaller than 16. The JX6-CON1 module has value 16 as ID.

Register 61568	Is assigned to module slot 2.
Register 61569	Is assigned to module slot 3.
Register 61570	Is assigned to module slot 4.
Register 61571	Is assigned to module slot 5.
Register 61572	Is assigned to module slot 6.
Register 61573	Is assigned to module slot 7.
Register 61574	Is assigned to module slot 8.

32.18 Specifying the floating point register array

Register 61810: Specifying the floating point register array	
Function	**Description**
Read	Start of the floating point register array
Write	Specifying the start of the floating point register array • with a value less than 1000 the value 1000 is written into the register • with a value less than 20479 the value 20479 is written into the register
Value range	1000 through 20479
Value after reset	20479

Meaning of register 61810:

Specifies the register where the floating point register array starts. The floating point register area always ends at 20479.

Example 1: The numeric value of register 61810 is 1000.

The floating point register array starts at register 1000 and ends at register 20479.

Example 2: The numeric value of register 61810 is 20479.

No floating point register array has been defined.

Switching over between integer and floating point registers/variables

The remanent variables/registers in the address range from 1000 through 20479 can be used as integer variables/registers, or as floating point variables/registers. In the special register 61810 the user can set the variable's/register's address number starting from which variable/register contents are regarded as floating-point numbers.

Example:

User variables/registers starting from number 3500 are to be treated as floating point variables/registers.

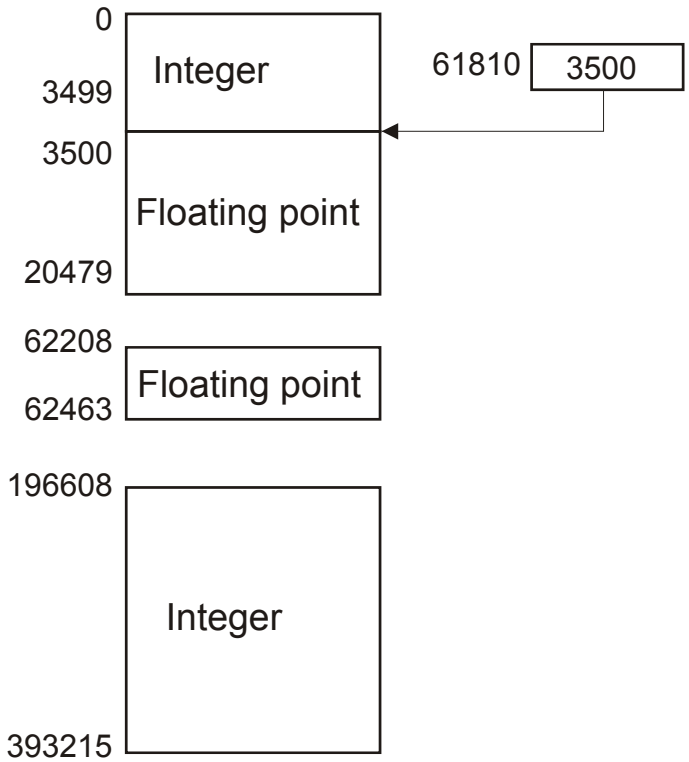


Fig. 55: Partitioning of integer/floating point registers

Note!

When "switching" between integer/floating point variables/registers, or vice versa, the contents of the memory cells remain unchanged. Merely the kind how they are interpreted changes.

33 Special Functions

33.1 General Information

JetSym

The special functions always are called using two parameters. The first parameter is the number of the first register of a register array which contains the operands (this array may, of course, consist of one register only). The second parameter is the number of the register into which the result of the function has to be written, for example:
The register numbers can be specified directly or indirectly.

This calling convention applies to all system functions except for system functions **60**, and **61**.

Example:

```
SPECIALFUNCTION (21, 9000, 9001)
```

Result:
This function calculates the sine of the number in register 9000 and stores the result in register 9001.

JetSym ST

These system functions are always called using two parameters. The first parameter is the address of a variable array which contains the operands (this "array" may, of course, consist of a single variable only).
The second parameter is the address of the variable into which the results of the function have to be written.
These addresses can be specified directly or indirectly.

This calling convention applies to all system functions except for system functions **60**, and **61**.

Example:

```
SYSTEMFUNCTION (21, &fSource, &fDestination);
```

Result:
This function calculates the sine based on the content of the variable 'fSource', and stores the result in 'fDestination'.

33.2 Function Types (Overview)

4	BCD -> Hex
5	Hex -> BCD
20	Square Root
21	Sine
22	Cosine
23	Tangent
24	Arc Sin
25	Arc Cosine
26	Arc Tangent
27	Exponential Function
28	Natural logarithm
29	Absolute value
30	MOD (modulo)
50	Sorting data
60	Calculating the CRC-Checksum
61	Checking the CRC-Checksum
65	Modbus/TCP; reading registers
66	Modbus/TCP; writing registers
80	Configuring RemoteScan
81	Starting RemoteScan
82	Stopping RemoteScan
90	Data files; saving values – creating a file
91	Data files; saving values – appending to a file
92	Data files; reading values from a file
96	Data files; deleting a file
110	Sending e-mail

33.3 Format Conversion

Special Function 4: BCD to Binary (no sign)

Function 4: BCD -> HEX Conversion	
Value range of argument:	BCD number with four decimal places (binary coded fractional number)
Value range of the result:	Binary number in hexadecimal format
Computing time:	Approx. 5 µs

Meaning:

This function can be used, for example, to convert multi-digit BCD switches, which are connected to digital inputs, into internal (binary) notation.

Special Function 5: Binary to BCD

Function 5: HEX -> BCD conversion	
Value range of argument:	Binary number in hexadecimal format
Value range of the result:	BCD number with four decimal places
Computing time:	Approx. 5 µs

Meaning:

This function can be used, for example, to convert a numeric value for output on a BCD display, which is connected to digital inputs.

33.4 Arithmetic Functions

Note!

On principle, it is permitted to specify integer registers for parameter transfer or for the result.
In most cases, this makes no sense due to the value range.

Special Function 20: Square Root

Function 20: Square Root	
Value range of argument:	0 and positive numbers
Value range of the result:	0 and positive numbers
Potential errors:	Negative number as argument
Result in case of error:	1.00
Computing time:	Approx. 5 µs

Special Function 21: Sine (sin)

Function 21: Sine	
Value range of argument:	-1000 through +1000 (radian measure!)
Value range of the result:	-1.00 through +1.00
Potential errors:	None
Computing time:	Approx. 5 µs

Special Function 22: Cosine (cos)

Function 22: Cosine	
Value range of argument:	-1000 through +1000 (radian measure!)
Value range of the result:	-1.00 through +1.00
Potential errors:	None
Computing time:	Approx. 5 µs

Special Function 23: Tangent (tan)

Function 23: Tangent	
Value range of argument:	-1000 through +1000 (radian measure!)
Value range of the result:	-1E13 to +1E13
Potential errors:	None
Computing time:	Approx. 5 µs

Special Function 24: Arc Sine (arc sin)

Function 24: Arc Sin	
Value range of argument:	-1.00 through +1.00
Value range of the result:	-pi/2 to +pi/2
Potential errors:	Argument outside -1.00 .. +1.00
Result in case of error:	1.00
Computing time:	Approx. 5 µs

Special Function 25: Arc Cosine (arc cos)

Function 25: Arc Cosine	
Value range of argument:	-1.00 through +1.00
Value range of the result:	-pi/2 to +pi/2
Potential errors:	Argument outside -1.00 .. +1.00
Result in case of error:	1.00
Computing time:	Approx. 5 µs

Special Function 26: Arc Tangent (arc tan)

Function 26: Arc Tangent	
Value range of argument:	-1E13 to +1E13
Value range of the result:	-pi/2 to +pi/2
Computing time:	Approx. 5 µs

Special Function 27: Exponential function (e^x)

Function 27: Exponential function (e^x)	
Value range of argument:	-30 through +30
Value range of the result:	1E-13 through 1E13
Potential errors:	Argument outside -30 .. +30
Computing time:	Approx. 5 µs

Special Function 28: Natural Logarithm (ln)

Function 28: Natural Logarithm (ln)	
Value range of argument:	1E-13 through 1E13
Value range of the result:	-30 through +30
Potential errors:	Argument < 1E-13
Computing time:	Approx. 5 µs

Special Function 29: Absolute value

Function 29: Absolute value	
Value range of argument:	Any value
Value range of the result:	Value (of argument)
Potential errors:	None
Computing time:	Approx. 3 µs

Special Function 30: Modulo

Function 30: MOD	
Value range of argument:	Any value
Value range of the result:	Value (of argument)
Value range of result 2:	Decimal positions (argument)
Potential errors:	None
Computing time:	Approx. 3 µs

33.5 Sorting data

Special Function 50: Sorting data

JetSym `SPECIALFUNCTION (50, Parameter1, Parameter2)`

JetSym ST `SYSTEMFUNCTION (50, Parameter1, Parameter2);`

Operating principle:

Special function 50 is used for sorting data within the controller.

For more information on sorting data and special function 50 refer to chapter 34 "Sorting data" on page 345.

33.6 Modbus RTU CRC-Checksum

Special/system functions 60 and 61 are for calculating and checking CRC checksums with a Modbus communication in RTU mode.

Special Function 60: Calculating the CRC-Checksum

Operating principle	This function calculates a two-byte checksum from the transferred frame and adds the two bytes to the end of the frame.
Parameter 1	Number of the register with the first datum of the MODBUS protocol.
Parameter 2	Number of the register with the last data of the MODBUS protocol without the two bytes for the CRC-checksum.
Potential errors	The number of the last register is smaller than the number of the first register. Each register may contain useful data in the lowest 8 bits only.
Result in case of error	Undefined
Computing time	Approx. 68 µs when the data length is 100 registers

Special Function 61: Checking the CRC-Checksum

Operating principle	This function checks the checksum of a frame and adds the result to the end of this frame.
Parameter 1	Number of the register with the first datum of the MODBUS protocol.
Parameter 2	Number of the register with the last datum of the MODBUS protocol with the two bytes for the CRC-checksum.
Potential errors	The number of the last register is smaller than the number of the first register. Each register may contain useful data in the lowest 8 bits only.
Result in case of error	Undefined
Computing time	Approx. 68 µs when the data length is 100 registers

JetSym sample program for checking a Modbus CRC checksum

In the following example, a Modbus CRC checksum is calculated and checked using special functions 60 and 61. If the checksum is correct, a "1" is returned. Else, a "0" is returned.

```
REGISTER_LOAD (100, 0x02)       // Slave Address
REGISTER_LOAD (101, 0x03)       // Function code
REGISTER_LOAD (102, 0x00)       // Starting number
REGISTER_LOAD (103, 0x20)       // Starting number
REGISTER_LOAD (104, 0x00)       // Quantity
REGISTER_LOAD (105, 0x04)       // Quantity
REGISTER_LOAD (106, 0x45)       // CRC checksum
REGISTER_LOAD (107, 0xF0)       // CRC checksum
SPECIALFUNCTION (61,100, 107)   // Checking CRC checksum
IF
     Reg. 108 = 1               // The result is contained in
                                   register 108
THEN ...                        // CRC checksum correct
ELSE ...                        // CRC checksum incorrect
```

33.7 Modbus/TCP

The special/system functions 65 (reading registers) and 66 (writing registers) can be used as acyclic transmission channel to a Modbus/TCP server.
These functions establish a connection to the specified server, transmit the desired data and clear down the connection.

Special Function 65: Reading registers

JetSym	`SPECIALFUNCTION (65, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (65, Parameter1, Parameter2);`

Operating principle:

Using this function, a register block from a Modbus/TCP server is copied to the registers/variables of the local memory.

Special Function 66: Writing to Registers

JetSym	`SPECIALFUNCTION (66, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (66, Parameter1, Parameter2);`

Operating principle:

Using this function, the content of registers/variables of the local memory is copied to a register block a Modbus/TCP server.

For more information on Modbus/TCP (client and server), as well as on special functions 65 and 66 refer to chapter 28 "Modbus/TCP" on page 229.

33.8 RemoteScan

The configurable RemoteScan function is used to cyclically copy register/variable contents from the JetControl to registers/variables of network nodes. On the other hand, they can be read by the nodes and copied into JetControl registers/variables.

The RemoteScan function can be accessed by using special/system functions 80, 81, and 82.

At the moment, RemoteScan via Modbus/TCP is supported only.

Special Function 80: Configuring RemoteScan

JetSym	`SPECIALFUNCTION (81, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (81, Parameter1, Parameter2);`

Operating principle:

To configure RemoteScan function 80 is used. However, this function does not start communication.

Special Function 81: Starting RemoteScan

JetSym	`SPECIALFUNCTION (81, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (81, Parameter1, Parameter2);`

Operating principle:

Function 81 is used to start a RemoteScan that has been configured using function 80.

Special Function 82: Stopping RemoteScan

JetSym	`SPECIALFUNCTION (81, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (81, Parameter1, Parameter2);`

Operating principle:

Function 82 is used to stop a running RemoteScan. When doing so, all possibly existing communication connections are closed.

For more information on RemoteScan, as well as on special functions 80, 81 and 82 refer to chapter 27 "RemoteScan" on page 225.

33.9 Data Files

Special Function 90: Saving values – Creating a file

JetSym	`SPECIALFUNCTION (90, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (90, Parameter1, Parameter2);`

Operating principle:

Function 90 is for creating a new data file and inserting a selectable register or flag block into this file.

Special Function 91: Saving values – Appending to a file

JetSym	`SPECIALFUNCTION (91, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (91, Parameter1, Parameter2);`

Operating principle:

Function 91 is for appending a selectable register or flag block to an existing file. In case this file does not exist, a new file will be created.

Special Function 92: Reading values from a file

JetSym	`SPECIALFUNCTION (92, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (92, Parameter1, Parameter2);`

Operating principle:

Function 92 is for reading register values and flag states out of a data file and entering them into the corresponding registers or flags. The information is processed in the order specified by the content of the file.

Special Function 96: Deleting file

JetSym	`SPECIALFUNCTION (96, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (96, Parameter1, Parameter2);`

Operating principle:

Function 96 is for deleting a data file from the flash disk.

For more information on data files, as well as on special functions 90, 91, 92 and 96 refer to chapter 29 "Data Files" on page 237.

33.10 E-Mail

Special Function 110: Sending an E-Mail

JetSym	`SPECIALFUNCTION (110, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (110, Parameter1, Parameter2);`

Operating principle:

This function is used to initiate the sending of an e-mail.

For more information on sending e-mails refer to chapter 25 "E-Mail" on page 201, as well as on special function 110 refer to chapter 25.4 "Sending E-Mails" on page 204.

34 Sorting data

To sort data within the controller, so far, it was necessary to program the sorting algorithm in SYMPAS, JetSym, or JetSym ST.
The advantage of this approach was that each source line was known. However, this approach had an adverse effect on the sorting performance.

To offer the user the greatest possible flexibility, the sorting algorithm is now separated from the data and is embedded in the OS of the controller. The data to be sorted are indirectly addressed through a descriptor using parameter 1.

34.1 Special/System Function 50

JetSym	`SPECIALFUNCTION (50, Parameter1, Parameter2)`
JetSym ST	`SYSTEMFUNCTION (50, Parameter1, Parameter2);`

Operating principle: Special/systemfunction 50 is used for sorting data which have been specified by parameter 1.

Parameter 1: Pointer to the "Input Descriptor".

Input Descriptor

Offset	Meaning	
0	Beginning of the data block with the data to be sorted	Number of the start register: 0 .. 20479
1	Quantity of data records	2 .. 1000
2	Data record length	2 .. 240
3	Sorting element	0 .. 999
4	Sorting mode	0 .. 1
	Bit 0 = 0: Ascending order	
	Bit 0 = 1: Descending order	
	Bit 1 to 3: Unassigned	
5	Unassigned	
6	Unassigned	
7	Unassigned	
8	Unassigned	
9	Unassigned	

Parameter 2: Number of the register / address of the variable resulting from the function.

Offset	Meaning	
0	Error code	
1	Execution time in µs	100 data records are sorted in approx. 4000 microseconds
2	Unassigned	
3	Unassigned	

Potential errors:

Error code	Meaning
0	OK, no error
1000	Star tregister > 20479
2000	Registers for return value > 20479
3000	Quantity of data records > 1000
4000	Data length > 1000
5000	Sorting mode > 256
6000	Sorting element > data length minus 1

Important!

Before calling the function, the input descriptor has to be initialized using values that make sense.

34.2 JetSym Sample Program

```
TASK 1
// Sorting data using special function 50
// Writing the input descriptor
     REGISTER_LOAD (100, 400)     // Specifying the register where
                                  the data to be sorted are located
                                  The data block with the data to
                                  be sorted starts with register
                                  400.
     REGISTER_LOAD (101, 4)       // Quantity of data records
     REGISTER_LOAD (102, 3)       // Data record length:
                                  3 elements
     REGISTER_LOAD (103, 1)       // Element 2 serves as sorting
                                  criterion
     REGISTER_LOAD (104, 0)       //Sorting in ascending order
                                  // On completion, the least value
                                  is contained in register 401.
     REGISTER_LOAD (105, 0)       // No function
     REGISTER_LOAD (106, 0)       // No function
     REGISTER_LOAD (107, 0)       // No function
     REGISTER_LOAD (108, 0)       // No function
     REGISTER_LOAD (109, 0)       // No function

// Before sorting, the data block contains the following numerals
     REGISTER_LOAD (400, 1)
     REGISTER_LOAD (401, 8)
     REGISTER_LOAD (402, 15)
     REGISTER_LOAD (403, 2)
     REGISTER_LOAD (404, 1)
     REGISTER_LOAD (405, 8)
     REGISTER_LOAD (406, 3)
     REGISTER_LOAD (407, 4)
     REGISTER_LOAD (408, 20)
     REGISTER_LOAD (409, 4)
     REGISTER_LOAD (410, 20)
     REGISTER_LOAD (411, 6)

// Sorting data using special function 50
     SPECIALFUNCTION (50, 100, 200)
     IF
          REG 200 # 0             // Error code unequal to 0?
        THEN
          DISPLAY_TEXT (0, 20, "Error")
          GOTO sError
        ELSE
     .....
```

Fig. 56 shows the data block before the sorting process. Fig. 57 shows the data block after the sorting process.

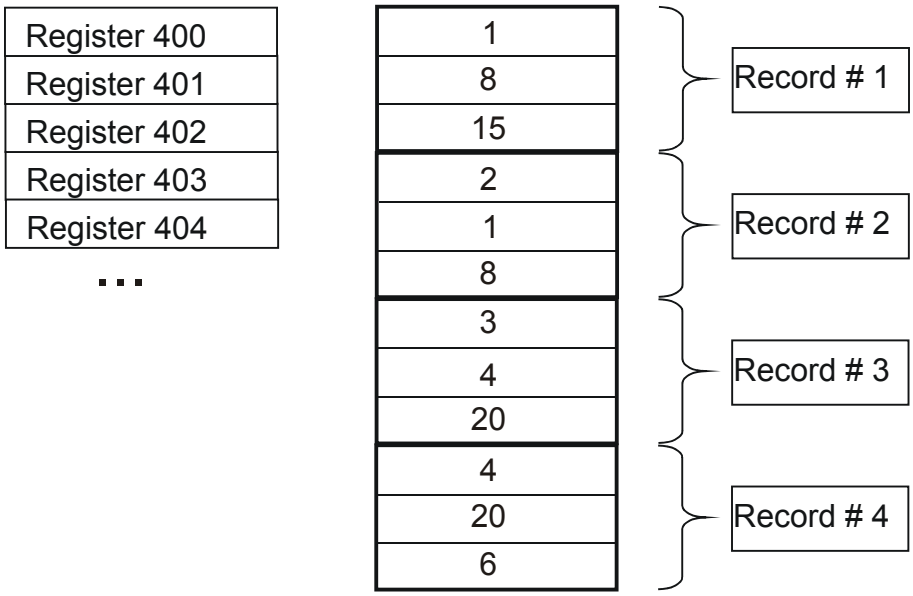


Fig. 56: Data block before the sorting process

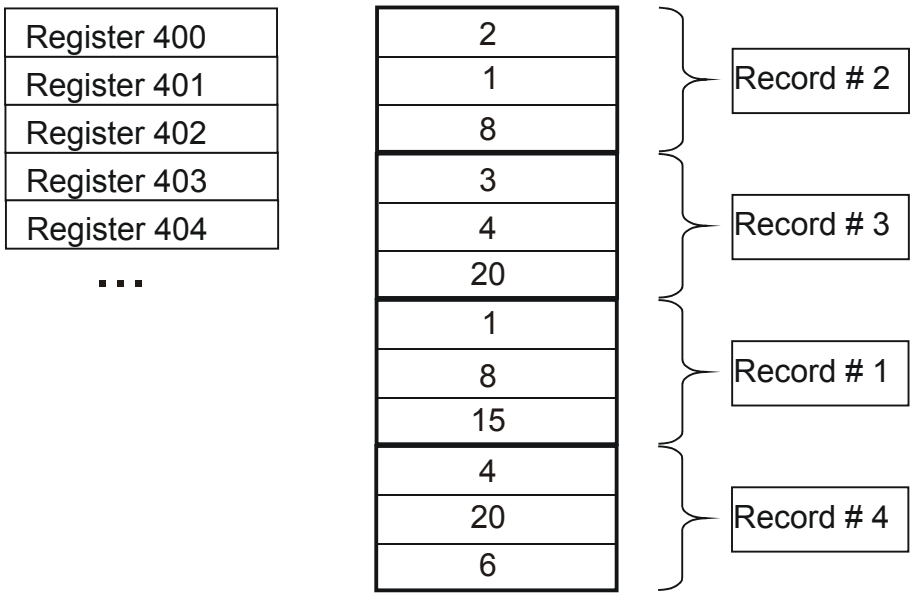


Fig. 57: Data block after the sorting process

35 Oscilloscope of JetSym

The values of user registers 0 through 20479 can be represented in the oscilloscope window of the JetSym program.

The online help of JetSym describes the possibilities of this window and what needs to be observed when representing register values in the oscilloscope window.

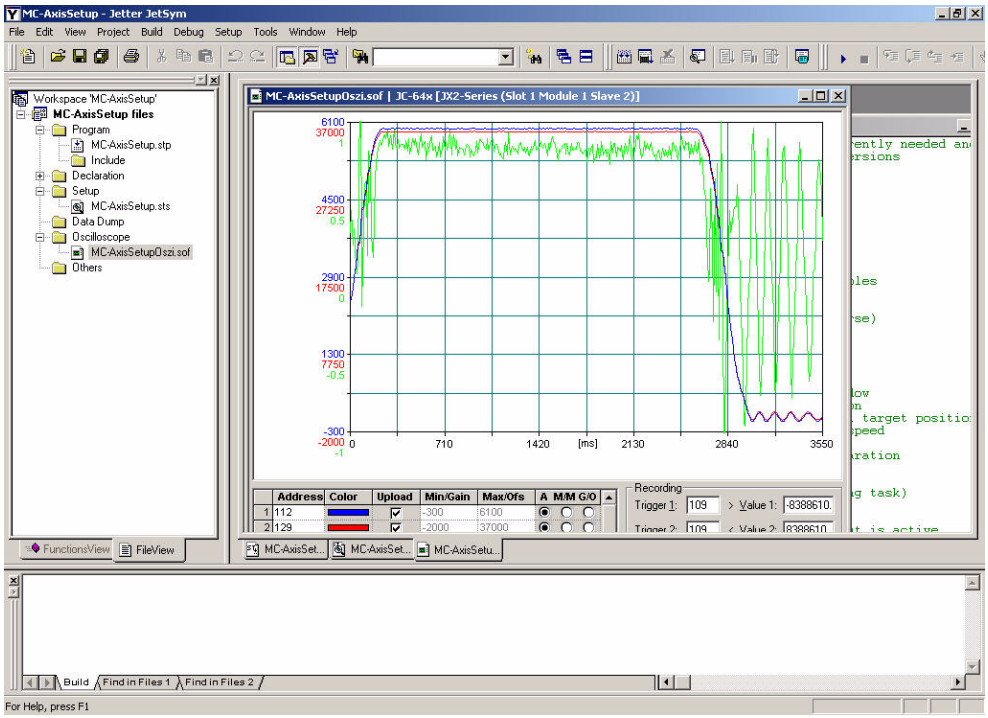


Fig. 58: JetSym Screen

36 User-programmable Interface

36.1 Description of Connections

See chapter 15.6 "User-Programmable Interface" on page 99.

36.2 Register Description

See chapter 32.12.3 "User-Programmable Interface" on page 315. The configuration register is described in chapter 32.12 "Controlling Network # 2" on page 312.

36.3 Program example

36.3.1 Program Listing

```
// The program receives the ASCII characters „A" to „Z"
// and sends them back via the PRIM interface converted into
// lower-case letters

DEF_FUNCTION (EmpfPrim, EP)
Par: rFirstChar, rLastChar
Var: rHelp

// Function for reading out a character from the receive buffer
     REGISTER_LOAD (rHelp, @rPRIMEmpf)  // Read character from
                                        // buffer and check
                                        // character for
                                        // valid range
     IF LIMITS (RecPrim, @rFirstChar, @rLastChar)
     THEN
     REGISTER_LOAD (RecPrim, @rHelp
     ELSE
     REGZERO RecPrim
     THEN
   RETURN
END_DEF

DEF_FUNCTION (SendPrim, SE)
Par: rSendChar

// Function for writing a character into the transmit buffer

     WHEN REG rSendCnt < 128 THEN
     REG rPRIMSend = REG rSendChar + 32
     THEN
   RETURN
END_DEF



TASK tPRIMhandling
```

```
    REG rPRIMconfig =                   // RS422 configuration
    0b00000000000000000010111000        // even parity
                                        // 8 data bits
                                        // 1 stop bit
    REGISTER_LOAD (rPRIMBaud, 7)        // 19200 baud

  LABEL mPRIMLoop
    WHEN NOT REGZERO rRecCont THEN      // Are there any
                                        // incoming characters?
      REG rChar = RecPrim (90, 65)
    IF REGZERO rChar THEN               // Is there a valid
                                        // character?
      GOTO mPRIMLoop                    // No
    ELSE
      SendPrim @rChar
      REGZERO rChar                     // Acknowledge
      GOTO mPRIMLoop
```

36.3.2 Symbol Listing

```
1     tPRIMHandling      0          // Task
2
3     mPRIMLoop          !          // Label with "!" indicates
                                    // that to the label no
                                    // number has been assigned.
4     rPRIMconfig        62989
5     rPRIMBaud          62990
6     rPRIMSend          62992      // Sending register
7     rPRIMReceive       62993      // Receiving Register
8     rReceivCnt         61499      // Receive buffer filling level
9     rSendCnt           61502      // Send buffer filling level
10    rChar              100
11
```

Note!

In the example above, sending and receiving of characters is divided into several functions:

- A character is sent if the value is written into the sending register.
- Occupancy of the the receive buffer is queried from register 61499.
- Accessing register 62663 deletes characters in the receive buffer.
- Occupancy of the the transmit buffer is queried from register 61502.

37 Real-Time Clock

With the help of a battery buffered register set access to the functions of the real-time clock is made.

Generally, hours are represented in 24 hour-format.
The days of the week are coded as follows: 1 = Sunday, 2 = Monday, 3 = Tuesday, etc.

Overview: Real-time clock registers for accessing individual registers	
Register #	**Function**
62920	Seconds
62921	Minutes
62922	Hours
62924	Day of the week 1 .. 7
62925	Day
62926	Month
62927	Year 0 .. 99

Overview: Real-time clock registers for accessing the register block (buffer)	
Register #	**Function**
62912	Seconds
62913	Minutes
62914	Hours
62916	Day of the week 1 .. 7
62917	Day
62918	Month
62919	Year 0 .. 99

- **Dummy write access to register 62928 writes the contents of register 62912 to register 62919 into the registers 62920 to 62927 at the same time.**
- **Dummy read access to register 62928 reads the content of registers 62920 through 62927 simultaneously into registers 62912 through 62919.**
- **Advantage: data consistency since all register contents are transfered at the same time.**

Example for dummy write access

```
REGISTER_LOAD (62928, 0)
```

Example for dummy read access

```
REGISTER_LOAD (28, @62928)
```

Sample Program for Real-Time Clock

The following sample program shows the present real-time clock data on the LCD 110 display.
The following approach is used to display minutes and seconds with a leading zero:

For right justified display of numbers it is possible to specifiy the number of digits to be displayed by using register `61453.` If less digits are allowed than there are significant digits in the number, then leading digits are suppressed.

The program realizes this approach by adding the value 100 to the seconds and minutes. Then, the leading "`1`" will not be displayed.

```
TASK 2
; Set display format: no sign, 2 digit numbers
     REGISTER_LOAD (61781, 1)               ; Without sign
     REGISTER_LOAD (61453, 3)               ; 2-digit number
     DISPLAY_TEXT (0, 1, "_Current time is:")

LABEL RTCloop
     CALL DisplayTime
     DELAY 5
GOTO RTCloop

LABEL DisplayTime
     REGISTER_LOAD (rI, @62928)             ; Dummy read access
     DISPLAY_TEXT (0, 24, ".  .20  ,  :  :")

     ; Displaying the date
     DISPLAY_REG (0, 22, 62917)             ; Day
     REG rI = REG 62918 + 100               ; Adding leading
                                            ; zeros
     DISPLAY_REG (0, 25, rI)                ; Month
     REG rI = REG 62919 + 100               ; Adding leading
                                            ; zeros
```

```
        DISPLAY_REG (0, 30, rI)                 ; Year

        ; Displaying the time
        DISPLAY_REG (0, 33, 62914)              ; Hour
        REG rI = REG 62913 + 100                ; Adding leading
                                                ; zeros
        DISPLAY_REG (0, 36, rI)                 ; Minute
        REG rI = REG 62912 + 100                ; Adding leading
                                                ; zeros
        DISPLAY_REG (0, 39, rI)                 ; Second
RETURN
```

38 Programming Language JetSym ST

Note!

Application programs can be programmed in the programming language JetSym ST starting from JetControl 647 operating system release 3.00 and JetSym Version 2.00.

Appendices

Appendix A: Downloading the Operating System

From JetSym

In the menu item "Build“ of the JetSym programming interface the operating system can be updated.
For this purpose, operating system files (*.OS) are made available on the Internet (http://www.jetter.de) by JETTER AG.

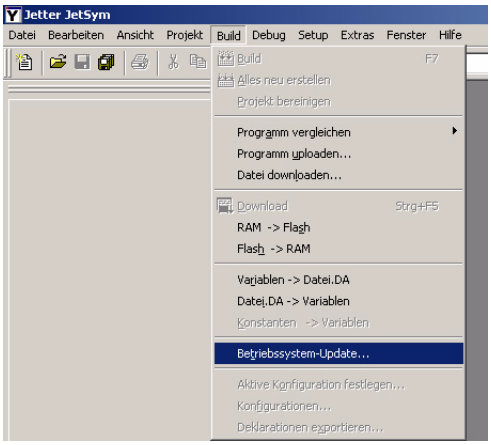


Fig. 59: Updating the OS File

For downloading an OS update, time-out must be set to 4000 ms in the JetSym menu "Project / Settings..." and there on the tab "Controller". This is the default setting. Furthermore, the baud rate U/D has to be set to 9600 baud.

In addition to this, the application program must be stopped during download of an OS update (by sliding the START/STOP switch in STOP position).

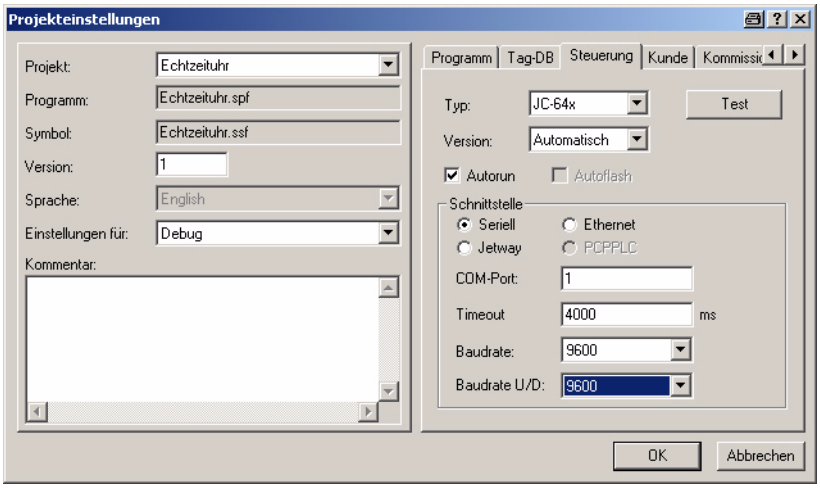


Fig. 60: JetSym Menu "Project Settings"

At the beginning of the OS update process via one of the **serical ports**, a special loader program is transferred to the controller and started there. This loader program is for receiving and storing an OS file sent by JetSym. It has no additional functions.

Note!

JetSym checks whether the selected OS file is compatible with the module to which the file is to be transferred.

Important!

Subject:JC-647 operating system update via Ethernet

- The new operating system will be activated only after the next boot process. For this purpose, the supply voltage (24 V) has to be switched off, and then reapplied.
- In case the OS update did not work (for example, because the connection has been interrupted, etc.), **avoid** at all costs to interrupt the 24 V power supply. Otherwise, only a minimum operating system (initial program loader) without Ethernet functions will be avaible.
- If the JC-647, for one reason or another, gets into initial program loader state, the operating system has to be transferred via serial interface at 9600 baud. This process takes quite a while.
- During an OS update via Ethernet interface, the controller continues to work maintaining its full range of functions (system bus, web functions etc.).

Appendix B:Recent Revisions

Chapter	Comment	Revised	Added	Deleted
Introduction	Significance of this User Manual	✓		
	History		✓	
	Description of Symbols		✓	
	Notes on OS and JetSym Version			✓
Chapter 1	Usage as Agreed Upon	✓		
	Who is Permitted to Operate the JC-647	✓		
	Repair & Maintenance	✓		
Chapter 1.1	Ensure your own safety	✓		
Chapter 1.3	Instructions on EMI	✓		
Chapter 2.1	Submodule listing	✓		
Chapter 2.2	Register areas	✓		
Chapter 2.3	Program memory		✓	
Chapter 7.1	Name of submodule JX6-ASI has been changed to JX6-AS-INT1	✓		
	Reference to additional information		✓	
Chapter 7.2	Module type has been changed from ASI-MODULE-TYPE to AS_Interface	✓		
Chapter 7.4	Submodule JX6-AD8		✓	
Chapter 7.5	Submodule JX6-DA4		✓	
Chapter 7.6	Submodule JX6-SV		✓	
	Submodule JX6-SV: Assignment of 15-pin SUB-D connector	✓		
Chapter 8	Optional PC card instead of socket # 3	✓		
Chapter 11.1	Input delay time	✓		

Chapter	Comment	Revised	Added	Deleted
Chapter 14.2	Reset value of command register 63794	✓		
Chapter 15	Front panel JC-647 with reference to serial port		✓	
Chapter 15.3	Chapter 15.5 „HMI LCD Interface“ has been moved to chapter 15.3 Description of additional HMI cables	✓ ✓		
Chapter 16	Front panel with reference to Ethernet port		✓	
Chapter 17.4	Example on Double Indirect Addressing	✓		
Chap. 17.4.1	JetSym ST instructions		✓	
Chap. 17.4.2	JetSym ST instructions		✓	
Chapter 18.1	Additional HMIs		✓	
Chapter 18.3	Multi-display mode		✓	
Chapter 18.4	- Cursor positions - JetSym ST instructions in examples		✓ ✓	
Chapter 19	Submodule JX6-SB(-I)	✓		
Chapter 19.4	Diagnostics by LED		✓	
Chapter 19.5	JETTER System Bus Cable	✓		
Chapter 19.6	Master/Slave Firmware Register area for Lumberg LJX7-CSL-... modules Register area for JetMove 2xx and JetMove 6xx servo amplifiers	✓	 ✓ ✓	
Chapter 19.6	Master-Slave Firmware	✓		
Chapter 19.7	Contents moved to chapter 19.1.2			✓
Chapter 19.8	Note has been moved to the beginning of chapter 19			✓
Ch. 19.10.1	Third-party modules	✓		

Chapter	Comment	Revised	Added	Deleted
Chapter 20.3	Instruction N_SEND_VALUE and N_GET_VALUE has been added		✓	
Chapter 20.4	No support of numbers in the 50,000 range	✓		
Chapter 21.1	Setting the IP address with special reference to registers 63816, 63817, 63818, 63819	✓		
Chapter 22	Introduction to Web functions	✓		
Chapter 23	File system		✓	
Chapter 24	FTP server		✓	
Chapter 25	E-mail	✓		
Chapter 26	HTTP server	✓		
Chapter 27	RemoteScan		✓	
Chapter 28	Modbus/TCP		✓	
Chapter 29	Data files		✓	
Chapter 30	Special flags for the keys of LCD 27 and LCD 16 Special flag 2112	 ✓	✓	
Chapter 31	Register offset 50,000 Registers 61696 - 61703 Register 63794: Activation of JETWay network only Reg. 185200 - 185299 RemoteScan registers Data file registers Register 61810: Specifiying the floating point register array	 ✓	 ✓ ✓ ✓ ✓	✓ ✓
Chapter 32.2	Computing performance	✓		
Chap. 32.4.4	Multi-display registers 61788 - 61799 and 61787		✓	
Chap. 32.4.5	Registers 61696 - 61793			✓

Chapter	Comment	Revised	Added	Deleted
Chapter 32.8	RemoteScan registers		✓	
Chapter32.9	Data file registers		✓	
Chapter 32.10	Hint on instruction 13: Activation of JETWay Network is no longer required Instruction registers	 ✓	✓	
Chapter 32.11.4	Offset register for the 50,000 range			✓
Chapter 32.14	Interface activity monitoring		✓	
Chapter 32.18	Register 61810: Specifiying the floating point register array		✓	
Chapter 33	Special functions	✓	✓	
Chapter 34	Special function 50		✓	
Chapter 35	JetSym oscilloscope	✓		
In chapter 15, chapter 35, and Appendix A	JetSym menus (release 2.3.1) have been added	✓		
Appendix A	OS download		✓	
Appendix E	Listing of special register in numeric order		✓	
Last page	Addresses	✓		

Appendix C: Glossary

Amplitude modulation	Simple method of impressing a signal on a carrier wave, in which the amplitude of the carrier is made to vary in proportion to the instantaneous value of the signal.
Analog	A parameter, e.g. voltage, which can be adjusted steplessly. In contrast to digital.
Bit-coded register	A register the individual bits of which can be set or reset.
Burst	Synonym: 1) Short period of intense activity on an otherwise quiet data channel. 2) Short isolated sequence of transmitted signals. 3) Fast transient interference.
Bus	A set of hardware lines (conductors) used for data transfer among the components of a computer system. Buses are characterised by the number of bits they can transfer at a single time. Distinction is made between serial bus systems (transmission of one bit at a time) and parallel bus systems (simultaneous transmission of a group of bits over separate wires).
Checksum	A calculated value that is used to test data for the presence of errors that can occur when data is transmitted between control systems. The checksum is calculated for a given chunk of data by sequentially combining all the bytes of data with a series of arithmetic or logical operations.
Client	E.g. client/server architecture A "client" is, for example, a program or task, that requests a service provided by another program -- for example, a word processor that calls on a sort routine built into another program. On a local area network or the Internet, a computer that accesses shared network resources provided by another computer (called a server).
Contact spacing	Standardized spacing between two contacts of a connector (male/female) adjacent to each other.
Counter	A counter acquires the changes of state (edge) of a digital signal. At each change of state, the counter will increment (increase) or decrement (decrease) a register value, which then will be evaluated by the controller.
Cycle time	The application program (task) is executed in cycles. The duration of one cycle is referred to as cycle time.
DC/DC converter	A DC to DC converter is a circuit which converts DC power from one voltage to a another. It is a special class of power converter. A DC/DC converter is used if, for example, the board requires 5 volts and the control system is supplied by a 24 V power supply unit.

Digital	Binary presentation of a parameter, e.g. time. This parameter in digital representation can be changed in given steps only, that is in binary mode. Contrasted with analog.
DIP switch	**D**ual-**i**n-line **P**ackage Switch
Domain Name	An address of a network connection in the format that identifies the owner of that address in a hierarchical format: server.organization.type.
Download	In communications, to transfer a copy of a file from a remote computer to the requesting computer, for example, an OS from a PC to a JetControl.
EC Low Voltage Directive	To be considered when using electric devices of a rated voltage between AC 50 and 1000 V and between DC 75 and 1500 V.
Electrical isolation	With electrical separation, the sensor ground is electrically isolated from the internal ground (GND) of the controller.
Electro-Magnetic Compatibility	Definition according to the EMC regulations: "EMC is the ability of a device to function in a satisfactory way in an electro-magnetic environment without causing electromagnetic disturbances itself, which would be unbearable for other devices in this environment."
Flags	1 bit storage position for intermediate results which are required for linkage purposes. The state of the bit is either 0 or 1.
Floating Point Notation	The floating point notation is also called exponential notation. It is a numeric format that can be used to represent very large and very small numbers. Floating point numbers are stored in two parts, a mantissa and an exponent. For example, 456000 is expressed as 456E3.
Integer	Also called »integral number« or »whole number«. A positive or negative integral number, e.G. 37, -50 or 764. In programming, »integer« stands for a data type representing whole numbers. Calculations with integers are considerably faster than calculations with floating point numbers. Therefore, integers are commonly used for counting and numbering procedures. Integers can have a leading sign (positive or negative) or be unsigned (positive). In addition to this, distinction is made between long and short integers depending on the number of bytes they occupy in the memory. Short integers comprise a smaller range of numbers (e.g. -32,768 to +32,767) than long integers do (for example -2,147,483,648 through +2,147,483,647).
IO Module Number	Number of a JX2-I/O module, a JX-SIO or another module connected to the JETTER system bus.

JETTER System Bus	A CAN bus which has been modified to meet JETTER's software and hardware requirements.
JX2-I/O Module	Non-intelligent JETTER system bus expansion module, such as JX2-ID8, JX2-OD8, ...
JX2-Slave Module	Intelligent JETTER system bus expansion module, such as JX2-SM1D, JX2-SV1, ...
Master/Slave Arrangement	A system in which one device, called the master, controls another device, called the slave.
Multitasking	A mode of operation offered by an operating system in which a computer works on more than one task at a time.
Parity	With control systems, parity refers to an error checking procedure where the number of 1s in each successfully transmitted set of bits must be an even or an odd number depending on the definition.
Pulse Width Modulation	The unmodulated signal consists of a continous sequence of pulses with constant frequency, pulse width and amplitude. During modulation, the pulse width is varied to reflect the information to be coded.
Real-time clock	A battery-backed integrated circuit chip in a control system that keeps track of the time.
Registers	A high-speed memory for a group of bits placed in a microprocessor or in another electronic device where data can be buffered for a specific purpose. With JETTER controllers, usually, these are 32 bit wide storage positions in a remanent RAM.
RemoteScan	States of digital inputs or outputs which are transmitted as frames via serial lines, CAN bus or PROFIBUS over distances from ten to several hundred meters.
Resolution	With D/A or A/D converters: The resolution may be expressed as the number of bits in the digital value that corresponds to a full-scale analog value. For example, the analog range may be a voltage ranging from -10 V to +10 V or a current ranging from 0 to 20 mA.
Ripple - Smoothing - Filtering	Ripple: The percentage of AC left on a DC signal after rectifying. Filtering: Circuit configuration with a RC or LC component in order to achieve more smoothness or a lower ripple of the DC voltage.

Sampling interval	An A/D converter requires a certain time, i.e. the sampling interval, to convert an analog value into a digital value. Within the sampling interval, a counter records changes in the digital input signal.
Semaphore	In programming, a signal -- a flag variable -- used to govern access to shared system resources.
Server/client arrangement	On the Internet or other network, a computer or program that responds to commands from a client. For example, a file server may contain an archive of data or program files. When a client submits a request for a file, the server transfers a copy of the file to the client.
Serverside Includes	A mechanism for including dynamic text in World Wide Web documents. Server-side includes are special command codes that are recognized and interpreted by the server. Their output is placed in the document body before the document is sent to the browser. With JETTER control systems, control variables are integrated into text.
Slave Module Number	Number of a JX2-Slave module connected to the JETTER system bus.
Tag	In markup languages such as SGML and HTML, a code that identifies an element in a document, such as a heading or a paragraph, for the purposes of formatting, indexing, and linking information in the document. For example, in HTML, <I>hello world</I> indicates that the phrase "hello world" should be italicized.
Tap Line	Open end of a line connected to the system bus.
TASK	A stand-alone application or a subprogram that is run as an independent entity.
Timeout	The amount of time the system will wait for a peripheral device to respond before it detects and reports this as an error.
Vibration Resistance	The device can permanently or shockwise be exposed to a vibration defined in the standard.

Appendix D: List of Abbreviations

AC	**A**lternating **C**urrent
AD	**A**nalog/**D**igital
AM	**A**mplitude **M**odulation
approx.	approximately
B	Width
betw.	between
CE	**C**ommunautés **E**uropéennes = European Union
cf.	compare
Cl.	Class
COM	Device name for a serial port in Wintel systems. The first serial port is identified as COM1, the second as COM2, etc.
CPU	**C**entral **P**rocessor **U**nit
D	Depth
DA	**D**igital/**A**nalog
DC	**D**irect **C**urrent
e.g.	[lat. **e**xempli **g**ratia] for example
EMC	**E**lectro **M**agnetic **C**ompatibility
EN	**E**uropäische **N**orm, that is: European Standard
ESD	**E**lectro**S**tatic **D**ischarge
EU	**E**uropean**U**nion
Fig.	Figure
FTP	**F**ile **T**ransfer **P**rotocol: Protocol used for copying files to and from remote computer systems on a network using TCP/IP, such as the Internet.
GND	**G**rou**nd** „Grounding System“
Gr.	Group
H	Height
HTML	**H**yper**T**ext **M**arkup **L**anguage: The markup language that uses tags to mark elements, such as text and graphics, in a document to indicate how Web browsers should display these elements to the user and should respond to user actions such as activation of a link by means of a key press or mouse click.
HTTP	**H**yper**T**ext **T**ransfer **P**rotocol: The client/server protocol used to access information on the World Wide Web.

i.e.	[Latin: **i**d **e**st] that is
IEC	**I**nternational **E**lectrotechnical **C**ommission
IP	**I**nternational **P**rotection
JetSym	Symbolic program processing language for Windows
JETWay	Serial system bus developed by Jetter AG for networking several Jetter controllers. In this bus system there is one master and a maximum of 98 slaves per hierarchical level.
LCD	**L**iquid **C**rystal **D**isplay
LED	**L**ight - **E**mitting **D**iode
max.	maximum
min.	minimum
neg.	negative
PC	**P**ersonal **C**omputer
PE	**P**rotective **E**arth, respectively "Protective Earth Conductor"
PID	**P**roportional/**I**ntegral/**D**ifferential (control)
PLC	**P**rogrammable **L**ogic **C**ontroller
POP3	**P**ost **O**ffice **P**rotocol version **3**: This is the current version of the Post Office Protocol standard in common use on TCP/IP networks.
pos.	positive
Reg.	Registers
resp.	respectively
RH	**R**elative **h**umidity
RM	Contact spacing
RTU	**R**emote **T**erminal **U**nit: Communication mode in Modbus networks via serial interfaces
SELV	**S**afe **E**xtra **L**ow **V**oltage: Voltage, which, under all operating conditions will not exceed a peak or DC voltage of 42.4 V. This voltage is either measured between two conductors or between one conductor and earth. The circuit, in which this voltage occurs, must be separated from the mains power supply by a safety isolating transformer or some equivalent.
SMTP	**S**imple **M**ail **T**ransfer **P**rotocol: A TCP/IP protocol for sending messages from one computer to another on a network. This protocol is used on the Internet to route e-mail.
specif.	specific
SSI	**S**ynchronous **S**erial **I**nterface

SUB-D	Type name of a plug-in connector
SYMPAS	Symbolic program processing language (**Sym**bolische **P**rogramm**a**blauf**s**prache)
TCP	**T**ransmission **C**ontrol **P**rotocol. The protocol within TCP/IP that governs the breakup of data messages into packets to be sent via IP, and the reassembly and verification of the complete messages from packets received by IP.
TCP/IP	Acronym for **T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol. A protocol for communications between computers which has become the de facto standard for data transmission over networks, including the Internet.
tr	Rise time of burst
tn	Total duration of burst
typ.	typical

Units:

A	Ampere (electric current)
dB	Dezibel
g	gram
Hz	Hertz
MHz	Megahertz (1 MHz = 10^6 Hz)
m	Meter
min	Minute
mm	Millimeter (1 mm = 10^{-3} m)
mm^2	square millimeter
s	Second
ms	Millisecond (1 ms = 10^{-3} s)
µs	Mikrosecond (1 µs = 10^{-6} s)
V	Volt (electric voltage)
W	Watt (electric active power)
Ω	Ohm (electric resistance)
kΩ	Kiloohm (1 kΩ = 10^3 Ω)
°C	Degrees centigrade (temperature unit)
%	Per cent

Appendix E: Register Overview by Numeric Order

In the column "R/W", the type of access to a parameter is identified:

R = Read
W = Write

Register Number	Name	R/W	Description
61440	Status of application program	R	Functional group: Status of application program Page 324
61445	PC-card detection	R	Functional group: PC-Card Page 59
61446	Status message PC-Card	R	Functional group: PC-Card Page 60
61448	Display language	R/W	Functional group: Indication Page 285
61449	Prioritised task		Functional group: Task control Page 253
61452	Decimal position for DISPLAY_REG	R/W	Functional group: Control of the HMIs Page 279
61453	Field width (DISPLAY_REG instruction)	R/W	Functional group: Control of the HMIs Page 278
61455	Field width (USER_INPUT instruction)	R/W	Functional group: Control of the HMIs Page 282
61461	"Delete to End Of Line" character	R/W	Functional group: Control of the HMIs Page 290
61462	"Delete Screen" character	R/W	Functional group: Control of the HMIs Page 291
61467	Task switch conditions	R/W	Functional group: Task control Page 276
61470	Interface activity monitoring, part 1	R/W	Functional group: Interface Activity Monitoring Page 319
61471	Interface activity monitoring, part 2	R/W	Functional group: Interface Activity Monitoring Page 320

Register Number	Name	R/W	Description
61472	Restrictions of monitor functions	R/W	Functional group: Restrictions for User Input Page 296
61473	Operating system error	R	Functional group: Operating system messages Page 271
61474	Multitasking control	R/W	Functional group: Task control Page 277
61477	Operating system error	R	Functional group: Operating system messages Page 272
61478	Error, operating system messages	R	Functional group: Operating system messages Page 273
61480 .. 61487	Mapping of keys on the HMIs	R	Functional group: Mapping of LEDs of the HMIs Page 297
61488 .. 61489	Mapping of LEDs on the HMI keys	R	Functional group: Mapping of LEDs of the HMIs Page 298
61499	Receive Buffer Occupancy	R	Functional group: User-Programmable Interface Page 315
61502	Transmit Buffer Occupancy	R	Functional group: User-Programmable Interface Page 315
61508	Interface status	R	Functional group: User-Programmable Interface Page 316
61511	Setting timeout time in milliseconds	R/W	Functional group: Control of network 2 JETWay Page 265
61529	Toggle switch position when switching on	R	Functional group: Status of the application program Page 324
61530	Task number of the OPC error	R	Functional group: Operating system messages Page 251
61568 .. 61575	Detected module	R	Functional group: General Registers Page 328
61585	Network response time	R	Functional group: Times Page 310

Register Number	Name	R/W	Description
61586	Processing time of network instruction	R	Functional group: Times Page 311
61588	Setting timeout time in milliseconds	R/W	Functional group: Control of network 1 JETWay Page 263
61589	Indirect network addressing	R/W	Functional group: Control of network 1 Page 309
61610	Highest task number in the program	R	Functional group: Task control Page 253
61645	Number of time registers	R	Functional group: Timer register Page 267
61648	Indirect Cursor Positions (DISPLAY_TEXT and DISPLAY_REG instruction)	R/W	Functional group: Control of the HMIs Page 291
61649	Indirect cursor position (USER_INPUT instruction)	R/W	Functional group: Control of the HMIs Page 292
61653	Display time for monitor functions	R/W	Functional group: Control of the HMIs Page 292
61667	Mapping of LEDs on the HMI keys	R	Functional group: Control of the HMIs Page 299
61672	Error messages during access to slave modules	R	Functional group: Operating system messages Page 274
61683	Text selection for DISPLAY_TEXT_2 instruction	R/W	Functional group: Control of the HMIs Page 293
61684	Bit code of occupied module slots	R	Functional group: General Registers Page 327
61688	Submodule type socket 1	R	Functional group: General Registers Page 48
61689	Submodule type socket 2	R	Functional group: General Registers Page 48
61690	Device ID from tuple 01 device info 1	R	Functional group: PC-Card Page 60
61691	Device size byte from tuple 01 device info 1	R	Functional group: PC-Card Page 60
61692	Submodule type socket 3	R	Functional group: General Registers Page 48
61773	Min. cycle time in 1 ms increments	R	Functional group: Task control Page 253

Register Number	Name	R/W	Description
61774	Max. cycle time in 1 ms increments	R	Functional group: Task control Page 253
61777	Cycle time of all tasks in 1 ms increments	R	Functional group: Task control Page 253
61781	Sign suppression	R/W	Functional group: Control of the HMIs Page 279
61783	Switching over to monitor display	R/W	Functional group: Control of the HMIs Page 293
61786	Divisor for the value output DISPLAY_REG	R/W	Functional group: Control of the HMIs Page 280
61787	Indirect Buffer Number for Device # 0	R/W	Functional group: Control of the HMIs Page 285
61788	Text buffer for HMI # 1	R/W	Functional group: Control of the HMIs Page 286
61789	Text buffer for HMI # 2	R/W	Functional group: Control of the HMIs Page 286
61790	Text buffer for HMI # 3	R/W	Functional group: Control of the HMIs Page 286
61791	Text buffer for HMI # 4	R/W	Functional group: Control of the HMIs Page 287
61792	Basic key flag number for HMI # 1	R/W	Functional group: Control of the HMIs Page 287
61793	Basic key flag number for HMI # 2	R/W	Functional group: Control of the HMIs Page 287
61794	Basic key flag number for HMI # 3	R/W	Functional group: Control of the HMIs Page 288
61795	Basic key flag number for HMI # 4	R/W	Functional group: Control of the HMIs Page 288
61796	Register number for controlling LEDs of HMI # 1	R/W	Functional group: Control of the HMIs Page 289
61797	Register number for controlling LEDs of HMI # 2	R/W	Functional group: Control of the HMIs Page 289
61798	Register number for controlling LEDs of HMI # 3	R/W	Functional group: Control of the HMIs Page 289
61799	Register number for controlling LEDs of HMI # 4	R/W	Functional group: Control of the HMIs Page 290

Register Number	Name	R/W	Description
61804	Task timeout time in 1 ms increments	R	Functional group: Task control Page 253
61806	User time base	R/W	Functional group: Timer register Page 326
61810	Specifying the floating point register array	R/W	Functional group: Specifying the floating point register array Page 329
61818	Max. time for USER_INPUT	R/W	Functional group: Control of the HMIs Page 294
61825	Number of characters per line	R	Functional group: Control of the HMIs Page 294
61826	Number of lines	R	Functional group: Control of the HMIs Page 295
61827	Number of characters	R	Functional group: Control of the HMIs Page 295
61848	Time base for START_TIMER, TIMER_END		Functional group: Timer register Page 267
61952	Application program runtime in seconds	R	Functional group: Operating system messages Page 252
61953	Controller runtime since reset in seconds	R	Functional group: Operating system messages Page 252
61954	Controller runtime in time base units	R	Functional group: Operating system messages Page 252
61955	Number of checksum errors when receiving via the network	R	Functional group: Control of network 1 - Error messages Page 264
61956	Controller runtime since reset in ms	R	Functional group: Operating system messages Page 252
61976	Suggested value for USER_INPUT	R/W	Functional group: Control of the HMIs Page 281
62150	Access error controller module: Slot number - 1	R	Functional group: Access to controller modules Page 252
62151	Access error controller module: Axis number - 1	R	Functional group: Access to controller modules Page 252

Register Number	Name	R/ W	Description
62152	Access error controller module: Register Number	R	Functional group: Access to controller modules Page 252
62912	Register for accessing the register block (buffer) Seconds	R	Functional group: Real-time clock Page 353
62913	Register for accessing the register block (buffer) Minutes	R	Functional group: Real-time clock Page 353
62914	Register for accessing the register block (buffer) Hours	R	Functional group: Real-time clock Page 353
62916	Register for accessing the register block (buffer) Day of the week	R	Functional group: Real-time clock Page 353
62917	Register for accessing the register block (buffer) Tag	R	Functional group: Real-time clock Page 353
62918	Register for accessing the register block (buffer) Month	R	Functional group: Real-time clock Page 353
62919	Register for accessing the register block (buffer) Year	R	Functional group: Real-time clock Page 353
62920	Register for accessing individual registers Seconds	R	Functional group: Real-time clock Page 353
62921	Register for accessing individual registers Minutes	R	Functional group: Real-time clock Page 353
62922	Register for accessing individual registers Hours	R	Functional group: Real-time clock Page 353
62924	Register for accessing individual registers Day of the week	R	Functional group: Real-time clock Page 353
62925	Register for accessing individual registers Tag	R	Functional group: Real-time clock Page 353
62926	Register for accessing individual registers Month	R	Functional group: Real-time clock Page 353

Register Number	Name	R/W	Description
62927	Register for accessing individual registers Year	R	Functional group: Real-time clock Page 353
62928	Dummy write access copies the contents of registers 62912 through 62919 to 62920 through 62927	R/W	Functional group: Real-time clock Page 353
62976	Current toggle switch position	R	Functional group: Status of the application program Page 325
62977	Operating system version * 100	R	Functional group: Operating system messages Page 252
62983	Baud rate programming interface	R/W	Control of the Programming Interface (PC) Page 318
62984	Baud rate of HMI (LCD)	R/W	Functional group: Control of the HMIs Page 300
62989	Configuration network 2	R/W	Functional group: Control of network 2 Page 312
62990	Baud rate network 2	R/W	Functional group: Control of network 2 Page 313
62991	Network number network 2	R/W	Functional group: Control of network 2 Page 313
62992	Transmission Buffer	R/W	Functional group: Control of network 2 Page 316
62993	Reception buffer Access removes character	R	Functional group: Control of network 2 Page 317
62994	Reception buffer Access does not remove character	R	Functional group: Control of network 2 Page 317
62995	Network number network 1	R/W	Functional group: Control of network 1 Page 308
62996	Baud rate network 1	R/W	Functional group: Control of network 1 Page 308
63020	RemoteScan protocol register	R	Functional group: RemoteScan function Page 305
63021	Amount of communication units	R	Functional group: RemoteScan function Page 305

Register Number	Name	R/W	Description
63022	Activity state	R	Functional group: RemoteScan function Page 305
63768	Counter Selection Register	R/W	Functional group: Counter Page 83
63769	Counter content	R	Functional group: Counter Page 82
63770	Numerator with division by 0	R	Functional group: Operating system messages Page 274
63771	Program address which has caused division by 0	R	Functional group: Operating system messages Page 275
63773	Network response time	R	Functional group: Control of network 2 Page 314
63774	Quantity of network errors	R	Functional group: Control of network 2 Page 314
63775	Error code of last network error	R	Functional group: Control of network 2 Page 315
63780	Error code of last network access	R	Functional group: Control of network 1 Page 311
63781	Quantity of network errors	R	Functional group: Control of network 1 Page 311
63784	Number of error messages in the fault memory since reset	R	Functional group: Operating system messages Page 251
63785	Selection of error message	R/W	Functional group: Operating system messages Page 251
63786	Contents of the error message selected by reg. 63785	R	Functional group: Operating system messages Page 251
63787	Current build level	R	Functional group: Operating system messages Page 275
63794	Instruction Register	R/W	Functional group: Instruction Register Page 82, Page 195, Page 307
63796	Computing speed	R/W	Functional group: Computing speed Page 275
63809	Divisor for value input USER_INPUT	R/W	Functional group: Control of the HMIs Page 282

Register Number	Name	R/W	Description
63810	Decimal positions for USER_INPUT	R/W	Functional group: Control of the HMIs Page 283
63827	State of Web function initialization	R	Functional group: Web-Function Initialization Page 202, Page 204, Page 303
63828	Own IP address	R/W	Functional group: E-Mail Page 261
63829	SMTP server IP address	R/W	Functional group: E-Mail Page 261
63830	POP3 server IP address	R/W	Functional group: E-Mail Page 261
63831	Port number SMTP server	R/W	Functional group: E-Mail Page 261
63832	Port number POP3 server	R/W	Functional group: E-Mail Page 261
63833	Processing status of the e-mail	R	Functional group: E-mail function Page 304
63834	Task number	R	Functional group: E-mail function Page 304
63835	Processing State	R	Functional group: Data Files Page 306
63836	Task number	R	Functional group: Data Files Page 306
63892	Start delay	R/W	Functional group: Timer register Page 326
63895	Time for timeout	R	Functional group: Ethernet control Page 301
63896	Processing time of network instruction	R	Functional group: Ethernet control Page 301
63897	Quantity of network errors	R/W	Functional group: Ethernet control Page 302
63898	Error code of last network access	R	Functional group: Ethernet control Page 302
63900	Gateway LSB	R/W	Functional group: Ethernet control Page 259
63901	Gateway 2SB	R/W	Functional group: Ethernet control Page 259
63902	Gateway 3SB	R/W	Functional group: Ethernet control Page 259

Register Number	Name	R/W	Description
63903	Gateway MSB	R/W	Functional group: Ethernet control Page 259
63904	Subnet mask LSB	R/W	Functional group: Ethernet control Page 259
63905	Subnet mask 2SB	R/W	Functional group: Ethernet control Page 259
63906	Subnet mask 3SB	R/W	Functional group: Ethernet control Page 259
63907	Subnet mask MSB	R/W	Functional group: Ethernet control Page 259
63980	Ethernet Monitoring Time (JetIP)	R/W	Functional group: Interface Activity Monitoring Page 322
63981	PC monitoring time (pcom5)	R/W	Functional group: Interface Activity Monitoring Page 322
63982	LCD monitoring time (pcom5)	R/W	Functional group: Interface Activity Monitoring Page 322
63983	Monitoring time NET 1 (JETWay)	R/W	Functional group: Interface Activity Monitoring Page 322
63984	Monitoring time NET 2 (JETWay)	R/W	Functional group: Interface Activity Monitoring Page 323
66000 .. 66999	RemoteScan register	R/W	Functional group: RemoteScan Page 232
185000 .. 185099	Task status	R	Functional group: Task control Page 277
185200 .. 185299	Task time register for DELAY		Functional group: Task control Page 253

Appendix F: List of Illustrations

Fig. 1: Earthing, JX6 rack 18
Fig. 2: Shielding of SUB-D connectors in conformity with EMC standards. 20
Fig. 3: Functional Diagram 23
Fig. 4: Front, Side and Top Views of the JC-647 25
Fig. 5: JX6-Basis-1 Rack 33
Fig. 6: JX6-Basis-2 Rack 34
Fig. 7: JX6-Basis-4 Rack 35
Fig. 8: JX6-Basis-8 Rack 36
Fig. 9: Top and Side Views of the JX6 Rack 39
Fig. 10: Front View of the JX6 Rack with Modules 40
Fig. 11: Wheel of keying pins provided with the control system 42
Fig. 12: Usage of keying pins 42
Fig. 13: Top view of the JXC-Basis-4 Rack with Keyed Connectors. 43
Fig. 14: RUN - STOP - LOAD Switch of the JC-647 Module 61
Fig. 15: Power Supply Terminal of JC-647 64
Fig. 16: Digital Input LEDs of the JC-647 66
Fig. 17: External circuit of the digital inputs 11 and 13 68
Fig. 18: Side View, Submodule JX6-I16 69
Fig. 19: Front View, Submodule JX6-I16 69
Fig. 20: LEDs of the submodule JX6-I16 71
Fig. 21: External circuit of the digital inputs 43 and 45 73
Fig. 22: Side View of the JX6-O16 Submodule 75
Fig. 23: Front View of the JX6-O16 Submodule 75
Fig. 24: LEDs of the JX6-O16 Submodule 77
Fig. 25: External circuit of the digital outputs 11 and 15 79
Fig. 26: Overview of Counters 81
Fig. 27: Serial Interfaces of the JC-647 module 85
Fig. 28: Setting the Baud Rate for the Serial Interface 88
Fig. 29: Menu Selection of JETWay 94
Fig. 30: JETWay-H PC Card 96
Fig. 31: Setting the Port Address in JetSym 97
Fig. 32: Ethernet Connection 101
Fig. 33: Ethernet Interface of the JC-647 module 101
Fig. 34: Ethernet Connection between PC and JetControl 102
Fig. 35: Ethernet Connection between JetControl and Switch 102
Fig. 36: Ethernet Connection Switch to switch 103
Fig. 37: REGISTER_LOAD with numeric parameters 109
Fig. 38: REGISTER_LOAD with symbolic parameters 109
Fig. 39: Assigning a constant to a variable (JetSym ST) 110
Fig. 40: Indirect and Double Indirect Addressing (JetSym) 110
Fig. 41: Indirect and Double Indirect Addressing (JetSym ST) 111
Fig. 42: Example for Double Indirect Addressing with JetSym 113
Fig. 43: Example of Register Arithmetic (JetSym) 115
Fig. 44: Example of Register Arithmetic (JetSym ST) 121
Fig. 45: Pin Assignment of the Connecting Cable for Several LCD HMIs 140
Fig. 46: Remote Arrangement on the JETTER System Bus 151
Fig. 47: Networking of several JC-647 controllers in Master/Master mode 153
Fig. 48: JETWay-H for Management Level 171
Fig. 49: JETWay-R for Process Level 173
Fig. 50: File system and server, services 185
Fig. 51: Root directory of the flash disk 193
Fig. 52: Files in the directory "System" 193

Fig. 53: Session using the Windows NT 4.0 FTP Client 199
Fig. 54: Monitoring of the LCD interface 323
Fig. 55: Partitioning of integer/floating point registers 330
Fig. 56: Data block before the sorting process 348
Fig. 57: Data block after the sorting process 348
Fig. 58: JetSym Screen 349
Fig. 59: Updating the OS File 361
Fig. 60: JetSym Menu "Project Settings" 361

Appendix G: Index

A

Address Number
- Digital Inputs 68, 106
- Submodule JX6-I16 73, 106
- Submodule JX6-O16 79, 106

Administrator 189, 192, 197

C

Communication unit 225

Conformance 232

Connection
- Digital Inputs 67
- Power Supply 64
- Submodule JX6-I16 72
- Submodule JX6-O16 78

Counter content 83

D

Description of Symbols 5

Design
- JC-647 Module 25
- Submodule JX6-I16 69
- Submodule JX6-O16 75
- Submodule JX6-SB(-I) 147

Disposal 16

E

Earthing procedure 18

Electrical Data
- Digital Inputs 65
- Submodule JX6-I16 70
- Submodule JX6-O16 76
- Submodule JX6-SB(-I) 147

Electrical Safety 28

EMC - Emitted Interference 29

EMI - Immunity to Interference 29

Environment 27

F

File Names
- Data Files 237

Flash Disk 194

Formatting 195

Function
- Absolute value 336
- Appending data to a DA file 342
- Arc Cosine 335
- Arc Sin 335
- Arc Tangent 336
- BCD -> HEX Conversion 333
- Calculating the CRC-Checksum 338
- Checking the CRC-Checksum 338
- Configuring RemoteScan 341
- Cosine 334
- Creating a DA File 342
- Deleting a DA File 343
- Exponential Function 336
- HEX -> BCD conversion 333
- MOD 337
- Modbus/TCP Server
  - Reading registers 340
  - Writing to Registers 340
- Natural logarithm 336
- Reading values from a DA file 342, 343
- Sine 334
- Sorting data 337
- Square Root 334
- Starting RemoteScan 341
- Stopping RemoteScan 341
- Tangent 335

Functional Data
- Digital Inputs 65
- Submodule JX6-I16 69
- Submodule JX6-O16 75
- Submodule JX6-SB(-I) 148

I

Information Signs 17

Input power 27

Installation Guide 39

Installation Procedure 41

Instruction Register 307

Intelligent Submodules 45

Interface
- Ethernet 101
- Serial 85

IP address 177

J

JETTER System Bus Cable 159
JETWay-H Cable 95
JETWay-R Cable 98

K

Keying Pins 42
Keys 192

L

LED
Digital Inputs 66
RUN - ERR1 - PC CARD 62
Submodule JX6-I16 71
Submodule JX6-O16 77
Locks 190

M

Maintenance 16
Malfunction 20
Malfunctions 17
Mechanical parameters 28
Modifications 16

N

Noise immunity 19
Non-intelligent submodules 45

O

Operating Parameters 27
Oscilloscope mode
User Registers 349
Overlay
Flags on registers 107, 297, 299
Registers and digital inputs 124
Registers and digital outputs 130
Overview: User Interfaces 137

P

Password
Flash Disk 195
Power Supply 63
Power Supply Unit 63
Procedure for removing the module 39
Programming Cable JN-PK 87
Programming Instruction
ACTUAL_POS 136
AXARR 136
DISPLAY_REG 144
DISPLAY_TEXT 141
DISPLAY_TEXT_2 141
FLAG 105
IN 105
N_COPY_FROM 182
N_COPY_TO 179
N_GET_REGISTER 175
N_SEND_REGISTER 174
OUT 105
POS 136
REG 105, 116
REGDEC 118
REGINC 117
REGISTER_LOAD 105, 109
REGZERO 117
SHIFT_LEFT 119
SHIFT_RIGHT 120
USER_INPUT 145
WHEN .. THEN 107
Protocol ID 232

Q

Qualified Staff 15

R

Repair 16
RUN - STOP - LOAD Switch 61

S

Select counter 82
ST Programming Instruction
ACTUAL_POS 136
AXARR 136
DEC 122
DISPLAY_TEXT 141
DISPLAY_TEXT_2 141
DISPLAY_VALUE 144
INC 121
POS 136
SHIFT_LEFT 123
SHIFT_RIGHT 123
USER_INPUT 145
Submodule-Type Registers 47
Suggestion for keying 43
System bus

Coding of IO instructions 169
Diagnostics by LED 155
Instructions on the Topology 150
Number of modules connected to the bus 148
Register Area
additional manufacturers 164
JetMove 2xx 168
JetMove 6xx 168
JX2-I/O Modules 163
JX2-Slave Modules 167
JX-SIO 164
Lumberg LJX7-CSL-..- 164
Special Register 162
Third-party modules 149

T

Transaction ID 232

U

Unit ID 232
Usage as agreed upon 15
Usage Other Than Agreed Upon 15
User Administration 189
User Interface
Control character 143
Cursor Position 141
Device Number 141
Text 142
User Interface Cable JN-DK-Xm 90
User Interface Cable KAY_0386-xxxx 91
User Interface Cable KAY_0533-0025 93

Jetter AG

Gräterstraße 2
D-71642 Ludwigsburg

Germany

Phone:	+49 7141 2550-0
Phone - Sales:	+49 7141 2550-433
Fax:	+49 7141 2550-425
Fax - Sales:	+49 7141 2550-484
Hotline:	+49 7141 2550-444
Internet:	http://www.jetter.de
E-Mail:	sales@jetter.de

Jetter Subsidiaries

Jetter Asia Pte. Ltd.

32 Ang Mo Kio Industrial Park 2
#05-02 Sing Industrial Complex
Singapore 569510

Singapore

Phone:	+65 6483 8200
Fax:	+65 6483 3881
E-Mail:	sales@jetter.com.sg
Internet:	http://www.jetter.com.sg

Jetter (Schweiz) AG

Münchwilerstraße 19
CH-9554 Tägerschen

Switzerland

Phone:	+41 719 1879-50
Fax:	+41 719 1879-69
E-Mail:	info@jetterag.ch
Internet:	http://www.jetterag.ch

Jetter USA Inc.

165 Ken Mar Industrial Parkway
Broadview Heights
OH 44147-2950

U.S.A.

Phone:	+1 440 8380860
Fax:	+1 440 8380861
E-Mail:	bschulze@jetterus.com
Internet:	http://www.jetterus.com